PIE: Writer
Word Processing System

Program by:
SOFTWEST

HAYDEN SOFTWARE
600 Suffolk Street, Lowell, Mass. 01853

PIE Writer software was developed by
Thomas Crosley of SOFTWEST, Sunnyvale, California.

The name PIE is a trademark of SOFTWEST.

Apple Computer, Inc., makes no warranties, either express or impiled, regarding the enclosed computer software package, its merchantability, or its fitness for any particular purpose.

Applesoft, Apple II, and Apple //e are trademarks or registered trademarks of Apple Computer, Inc.

Disclaimer – Neither Hayden Book Company, Inc., nor the author of this program makes any representation or warranties with respect to the contents herein contained, **and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose,** and expressly disclaims any and all liability for any special, incidental, or consequential damages resulting from any use of the programs contained herein. Further Hayden Book Company, Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such changes.

Limited Warranty - To the original purchaser only, Hayden Book Company Inc. warrants the magnetic diskettes on which the system is recorded to be free of defects in matrerials and faulty workmanship under normal service and use for a period of 90 days from the date the program is delivered. If, during this 90 day period, a defect in a diskette should occur, the diskette should be returned to Hayden Software Company, 600 Suffolk Street, Lowell, MA 01853. Hayden will replace the diskette without any charge to you provided that you have previously returned the enclosed Warranty Registration Card. Your sole and exclusive remedy in the event of a defect is limited to replacement of the diskette as provided above.

Important Notice – This is a fully copyrighted work and as such is protected under the copyright laws of the United States of America. According to these laws, consumers of copywritten material may make copies for their personal use only. Duplication for any other purposes whatsoever would constitute infringement of copyright laws and the offender would be liable to civil damages of up to $50,000 in addition to actual damages, plus criminal penalties of up to one year imprisonment and/or a $10,000 fine.

This product is sold for use on a single computer at a single physical location. Contact the publisher for information regarding licensing for use at multiple-workstation or multiple-computer installations.

Important Notice to Apple //e Users:

A PIE System Configuration option is to re-define the standard PIE Commands to take advantage of the //e dedicated function keys: the up and down arrows, the TAB key, and the DELETE key. Using these keys can make PIE Writer easier to use; however, to make them perform their correct function requires altering the standard PIE command key layout. This option is further explained in Chapter 1.

If you choose to select the //e keyboard layout, you will want to change the manual to reflect the fact that the keys have been changed. Replace the corresponding pages in the manual with the //e replacement pages found after the index. Don't throw out the replaced pages; simply exchange them.

Contents

1
Introduction to PIE Writer

HOW TO USE THIS MANUAL 2
BASIC CONCEPTS 2
- The Cursor 3
- The Window 3
- Beyond the Right-Hand Margin 4
- Scrolling 4
- Commands 5
- Default Values 5
- Final Suggestions 5

GETTING STARTED 6
- Backing Up Your PIE Writer Diskettes 7
- Booting the PIE Writer Diskette 7
- If Additional Help Is Needed 8
- System Configure 8
- Notes About The System Configure Program 11

2
The PIE Text Editor and Command Processor

HOW TO USE THE TEXT EDITOR 13
PIE TEXT EDITOR LESSONS FOR THE APPLE //E KEYBOARD 15e
- Lesson 1: The Cursor 15e
- Lesson 2: Home and Bottom Home 16e
- Lesson 3: Simple Cursor Movement 16e
- Lesson 4: The Status Line 17e

Lesson 5: Beyond the Right-Hand Margin 17e
Lesson 6: Tabs 18e
Lesson 7: Moving the Cursor to the Beginning or End of a Line 18e
Lesson 8: Scrolling Pages and Lines 19e
Lesson 9: Entering Upper and Lowercase Characters 19e
Lesson 10: Destructive Cursor Moves 20e
Lesson 11: Deleting a Character 20e
Lesson 12: Inserting a Character 21e
Lesson 13: Shifting Text Left or Right 21e
Lesson 14: Returning to the Beginning of a File; Leaving the Text Editor and Entering the Command Processor 22e
THE COMMAND PROCESSOR 22e
Saving the Text Buffer 23
Beginning a New File 23
Typing Modes 24
Saving a File 25
A Caution About File Lengths 26
Editing a File 27
Displaying Your Filename 27
Viewing the Catalog of Files 28
Loading a File 28
PIE Summary 29

3
The FORMAT Text Processor

LESSON 1: INTRODUCTION 31
Entering FORMAT Dot Commands 31
Displaying a File 33
Reading Formatted Text on the Screen 34
Other FORMAT Commands 35
LESSON 2: PRINTING A FILE 35
Transferring Between the Text Processor, the Text Editor, and the System Menu 35
Printing Hard Copy 36
LESSON 3: PARAGRAPH CONTROL 39
Standard Paragraphs 39
Other Paragraph Commands 39
One-Time Paragraph Indent 41
LESSON 4: DESIGNING A LETTER 41
Calculating Margin Widths 42
Setting Up the Left Margin 42

Line Length 43
Right Margin 43
The Heading and Date 44
Inside Address and Salutation 45
Body of Letter 45
Closing and Signature 46
Saving and Reviewing the Letter 47
LESSON 5: INDENTATION 48
Extract Indent 48
List and Outline Indent 48
LESSON 6: RUNNING TITLES, PAGE NUMBERS, MARGINS, AND FINISHING TOUCHES 53
Running Titles 53
Page Numbers 55
Margins 55
Finishing Touches 58

4
PIE and FORMAT Configure

PIE CONFIGURE 61
Printer Interface 62
Lowercase and Special Characters 63
Other PIE Editor Defaults 65
Program Disk Slot, Drive, Volume 66
Doubletime Spooler Option 67
Saving PIE Configure Options 68
FORMAT CONFIGURE 68
Printer Interface 69
Lowercase and Special Characters 70
Page Format 74
Top and Bottom Margins 75
Paragraphs and Filling 76
Program Disk Slot, Drive, Volume 77
Saving FORMAT Configure Options 77

5
Reference Section

PIE TEXT EDITOR FOR THE APPLE //E 79e
Simple Cursor Movement 80e
Cursor Jumps 80e
Tabbing 81e

Line and Window Scrolling 82e
Go To a Line 83e
Inserting and Deleting 84e
Splitting and Joining Lines 86e
Copying and Moving Lines 87e
Search and Replace 88e
Text Entry Modes 90e
Displaying the Help Screen 91e
Cursor-Defined Commands 92e
Entering Control Characters 95e
Exiting the Editor and Transferring to the Command Processor 96e
PIE COMMAND PROCESSOR 97
Control Commands 97
Input/Output Commands 99
Auxiliary Commands 102
FORMAT TEXT PROCESSOR 108
FORMAT DOT COMMANDS 109
Paragraphs 110
Layout 112
Indentation 115
Running Titles and Margins 116
Centering, Underlining and Boldface 118
File Switching and Form Letters 119
Interactive Input and Control 123
Character Translation and Definition 125
ERRORS AND ERROR MESSAGES 128
Reset Error 128
Recovering Text After Rebooting 128
Disk Errors 128

6
Advanced Topics

MAIL MERGE 131
Primary File 132
Data File 134
Formatted Output 135
Printing Labels from the Same Data File 137
USER INPUT DURING FORMAT 138
Physical Printer Adjustment 138
Altering FORMAT Command Variables 139
SHELL FILE STRUCTURES 140
Shell File Commands 141

SHIFT KEY MODIFICATION 141
TELECOMMUNICATION 142
Transmitting Text with a Communications Card 143
Receiving a File with the Communication Card 145
PIE Writer to PIE Writer Communication 145
ADDRESS TABLES 146
Page Zero Buffer Pointers 146
Important PIE Addresses 146
Important FORMAT Addresses 147
Addresses Common to PIE and FORMAT 148
FORMAT Limits 149
Command Table Modification 149
Custom Program Area 149
Custom Printer Driver 149

Appendix A
ASCII Character Codes

CONTROL CHARACTERS 152
PRINTING CHARACTERS 153

Appendix B
PIE WRITER: Changes from Previous Versions

CHANGES FROM VERSION 2.1 157
CHANGES FROM VERSION 2.0 158
PIE Command Processor 158
PIE Editor 159
FORMAT Text Processor 160
Other Changes 161

INDEX 163

1

Introduction to PIE Writer

The PIE Writer system turns your Apple II, Apple //e, or Franklin ACE personal computer into an easy-to-use word processor that can produce reports, letters, memos, and other business documents quickly and efficiently. The PIE Writer system is also widely used by writers, authors and others who compose and edit at the keyboard for its powerful editing and formatting capabilities in producing every variety of manuscript imaginable. As you type at the keyboard, PIE Writer displays what you write on a video screen. You can easily revise the content and organization of your document while viewing it on the screen. You can add, change, or delete words, phrases, sentences, and entire paragraphs.

The PIE Writer system remembers everything you type and files it away for future use. This lets you recall a document prepared earlier, update it while viewing it on the screen, and file away the new version.

Finally, the PIE Writer system lets you enter codes, along with your text, that command the computer to set indents, insert blank lines, center lines, begin new pages, and right-justify paragraphs, as well as many other word processing functions. These codes appear only on your video screen, not on your printed report. You can type sentence after sentence without concern for the document's final appearance, then review the entire report, edit it, enter layout codes, and print out as many copies of the perfect, final document as you want.

The PIE Writer system consists of a number of computer programs, all working together and contained on one diskette. For you, the user, there are only three programs you need to get started: the PIE Text Editor, the PIE Command Processor, and the FORMAT Text Processor. The Text Editor lets you enter and edit text so that

you can write error-free letters and reports. The Command Processor performs all the PIE Writer "housekeeping" functions, such as getting a file from the diskette into the computer so you can work with it and filing away the changes you make. The Text Processor controls how your document will look when it is printed.

HOW TO USE THIS MANUAL

This documentation has been organized so that both the beginner and the seasoned professional can use it easily.

The first three chapters of the manual provide all the information about PIE Writer you need to start producing perfect letters and reports.

Chapter 1 starts at the very beginning: how to get the system started, from turning on your computer to handling diskettes. The introduction progresses through such basic word processing concepts as what the "cursor" is and how it works, how to view text in the "window," and what "scrolling" means.

Chapters 2 and 3 introduce you, in order, to the PIE Text Editor, PIE Command Processor, and the FORMAT Text Processor.

In the last three chapters, PIE Writer's truly outstanding features are discussed in greater, more technical, detail.

Chapter 4 begins the more advanced material, intended for users who have mastered the basic features of PIE Writer. Here, you learn how to adjust the PIE Writer system to work with any printer, and how to modify the system to provide for specific, "permanent" document layouts.

Chapter 5 reviews all the basic PIE and FORMAT commands already covered and introduces additional, more powerful, commands that can be brought into use as you gain experience.

Chapter 6 describes advanced PIE Writer features of interest to word processing professionals and programmers. The experienced user learns how to customize form letters, how to transmit over telephone lines to another computer, and even how to have other software work with PIE Writer. Address tables are included.

The index gives you quick and easy access to all PIE Writer terms and topics.

Finally, a Reference Card lists all the PIE Writer commands, their meanings, and uses. The card also has a chart of suggested key labels, single words that serve as memory aids in learning the simple commands. You can use this handy card whenever you are "stuck" for the right command.

BASIC CONCEPTS

Before you start doing any word processing, there are a few key words and concepts you should know. It is important that you read through this chapter carefully and understand these concepts before you turn on your system.

The Cursor

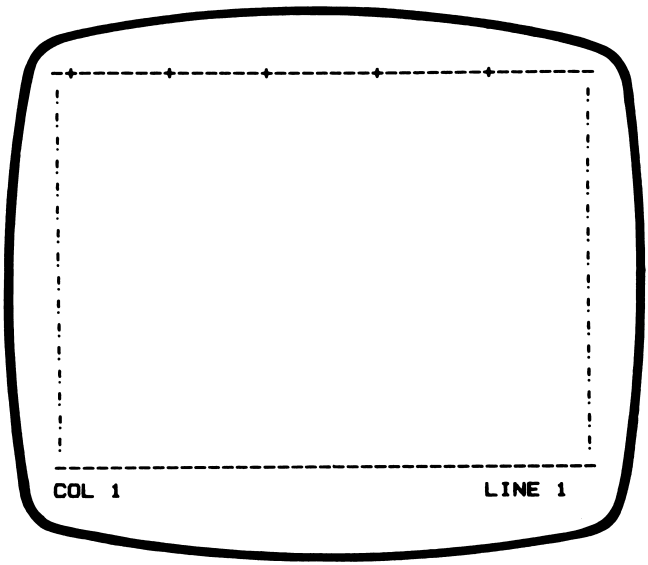
When you first use PIE Writer, a blinking rectangle, the cursor, will appear on your screen to the right of the phrase "WHICH ONE?". The cursor indicates your typing position on the screen, showing you exactly where your next typed character will appear.

When you get into the Text Editor program, the cursor will flash in the top left-hand corner of your screen. You can enter text from the keyboard that will appear on the screen at the cursor, or you may move the cursor to any other position on the screen and begin entering text at that point.

You will learn how to move the cursor around the screen in Chapter 2.

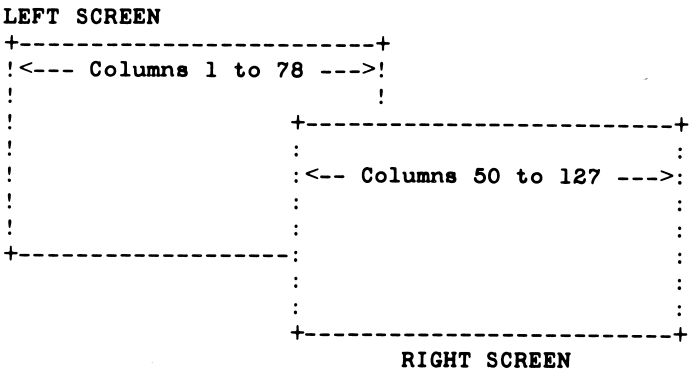
The Window

With the PIE Writer system, your video screen becomes a "window" through which you can use the text you have entered from the keyboard. The window can display 38 characters across the screen and 21 lines of characters down the screen. If your computer has an 80-column board on it, the window will display 78 characters across and 21 lines down. {**NOTE:** From now on, all references to hardware configurations other than a 40-column 48K Apple II Plus will be in brackets.}



Beyond the Right-Hand Margin

Although the window can show only 38 {or 78 for 80-column users} columns across at a time, you can still type in documents up to 64 {or 128} characters wide on the screen. The reason for this is that the window will shift automatically to right as you type the 39th {or 79th} character in a line, as shown below:



The right-hand side of the "page" reveals extra typing room.

When the window shifts to the right, you may feel lost for a moment because the right-hand side of the page partially overlaps the left-hand side. Characters 27 to 38 {or 51 to 78} from the left-hand side are displayed at the beginning of the line.

One last point about what you type and the confines of the window: No matter how wide or narrow you choose to type your lines on the screen, you can direct the PIE Writer system to print your document any width you desire. The window only restricts the number of characters you can see on the screen at once, not the final appearance of the document.

Scrolling

Most of the documents you type are likely to exceed 21 lines in depth. PIE Writer lets you type a document of any length.

Just as the window is capable of showing only 38 {78} characters across at a time, it can show only 21 lines of text at a time. The page showing in the window can be "scrolled" backward, toward the beginning of your document, or forward, toward the end of your document. This lets you view the entire document on the screen before you print it out.

Commands

There are three types of commands: single, double, and sequential. All the commands you type are shown in **bold** throughout the manual.

- *SINGLE KEY COMMANDS require that you press only one key in order to have the computer perform a function. Some single key commands will execute the function immediately once the key is pressed. Most single key functions, though, require that you press* **RETURN** *after entering the command before the computer will execute it.*
- *DOUBLE (or TRIPLE) KEY COMMANDS consist of two (or three) keys that must be held down at the same time. These commands are executed immediately, so the* **RETURN** *is not needed. The convention used to indicate a double key command is a hypen between the keys, as in* **CTRL-A**.
- *SEQUENTIAL COMMANDS use more than one key but do not require that the keys be held down together. The key shown first is typed first; the key shown second is typed second; and so on. In this manual, sequential commands are shown in the order they should be typed, without hyphens, as in* **ESC W RETURN**.

Single	Double	Sequential
E	**CTRL-A**	**ESC W RETURN**

Default Values

Default values are a series of specifications built into the PIE Writer system. The specifications represent values that most people would use in composing text. For example, if you tell PIE Writer that you want your paragraphs indented, but do not tell it how many spaces, it will indent 5 spaces automatically. If you want, you can "override" the default value by telling PIE Writer exactly how many spaces you want it to indent each paragraph.

In Chapter 4, you will learn how to set *new* PIE Writer default values to suit your own particular needs.

Final Suggestions

With these few basic concepts, you are now ready to begin using PIE Writer. Some final suggestions before you start:

1. *Do not be too concerned about making mistakes. PIE Writer can handle an infinite number of typographical and spelling errors. If you handle your diskettes correctly and insert them in the disk drive properly, you should have no mechanical problems with your diskettes, your disk drive, or your personal computer.*
2. *Keep your PIE Writer manual handy at all times.*

GETTING STARTED

To use this version of PIE Writer, you will need:

- *either:*
  an Apple //e computer, or
  an Apple II Plus (minimum 48K RAM), or
  an Apple II (with a language card), or
  a Franklin Ace 100 or 1000
- *the PIE Writer diskettes*
- *a disk drive and disk interface card*
- *a television set or monitor*
- *a printer and printer interface card*
- *extra blank diskettes*

The PIE Writer system assumes that you have all the equipment listed above. It also assumes, of course, that your equipment is installed correctly. It is important to note that:

- *The printer interface card should be in slot 1.*
- *The disk drive interface card should be in slot 6.*
- *An 80-column board, if you have one,* must *be in slot 3.*

If your system is not installed this way, see Chapter 4 on reconfiguring PIE Writer to work with your hardware.

PIE Writer is versatile, so it can work with almost any combination of printers, lowercase adapters, 80-column display boards, and 16K RAM (language) cards. If you have any difficulty installing your hardware, refer to the reference manuals for your computer or consult your local dealer.

Backing Up Your PIE Writer Diskettes

Before using PIE Writer, IT IS CRITICALLY IMPORTANT that you make a backup of each of your original PIE Writer diskettes. To make a backup, use either the COPYA or COPY program on your DOS 3.3 System Master diskette, as instructed in the DOS manual.

Once you have made a backup copy of each disk, store the original PIE Writer diskettes in a safe place. Then make one additional backup copy of the disk corresponding to the system you will be configuring *for*, either:

- *Disk 1: Apple II and Apple II Plus,* or
- *Disk 2: Apple //e and Franklin ACE.*

You should use this additional copy of the selected disk as your working system diskette, since the System Configure program, executed when you first boot a PIE Writer disk, will delete every file which is not needed for your current configuration.

Booting the PIE Writer Diskette

The first step in using the PIE Writer system is to learn how to "boot" your PIE Writer system diskette. This loads the first PIE Writer program file into the computer. To begin:

1. *Turn the computer power switch, located at the left rear of the computer, to OFF.*
2. *Take the PIE Writer system diskette in your hand (label facing upward, write-protect notch to the left, oval-shaped gap facing away from you)*
3. *Insert the diskette gently into your disk drive. Make sure that you insert your system diskette in drive 1 if you have more than one disk drive, or it will not boot.*
4. *Close the door of the drive.*
5. *Turn ON the computer, again using the switch at the left rear of the computer.*

The disk drive will whirr and click, and the disk drive's red IN USE light will glow. The PIE Writer diskette is now loading itself.

If Additonal Help Is Needed

If you should encounter any difficulty booting your system diskette, the problem may be solved by doing one of the following:

1. *If you do not have Autostart ROM, a feature in all but the oldest Apple II computers that allows you to boot a diskette more easily, you will see an asterik (*) on your video screen. This is a "prompt," meaning that the computer is waiting for you to give it additional instructions. When you see the **, type* **6 CTRL-P** *then press* **RETURN**, *and the diskette should boot.*

2. *If your diskette still does not boot, remove the diskette and restart the procedure for booting with a backup diskette. If this works, the first one you tried was damaged. Always be careful when handling your PIE Writer diskette. Mishandling can result in loss of information or permanent damage of the diskette.*

3. *If you're still at a loss about what's wrong, consult your local dealer or someone more familiar with diagnosing problems with your personal computer.*

System Configure

The first time you boot PIE Writer, you will be in the System Configure program. This *modifies* PIE Writer to work with your specific hardware configuration. IT IS CRITICALLY IMPORTANT that you make a backup copy of the PIE Writer diskette before working with the System Configure program. The System Configure program deletes files that are not needed for your present configuration. The original diskette should be stored safely. Use one of the backup copies to run the configuration program (by booting the diskette).

The next step is to confirm that you have made a backup when prompted by the System Configure program:

```
BEFORE CONTINUING, YOU SHOULD FIRST
MAKE A BACKUP COPY OF THIS DISK SINCE
SEVERAL FILES WILL BE DELETED

                HAVE YOU DONE THIS (Y/N)?
```

Type **Y** (for yes) to confirm. Next, the System Configure program determines the type of system you have and what optional hardware you have installed. It presents a list, such as:

```
CONFIGURE FOR

APPLE II PLUS

48K RAM

40 COLUMN

CONFIRM (Y/N)?
```

The system may be either APPLE //E, APPLE II PLUS, APPLE II (INTEGER), or FRANKLIN ACE. The memory size is either 48K or 64K (LANG CARD). The video display is either 40 COLUMN or 80 COLUMN (followed by the name of the 80-column board).

You are then asked to confirm the selection. If you answer **N** (for No), then the program will lead you through a series of prompts to select the particular system type, memory size, and video display you want to configure for. It then will display a new list and ask for confirmation.

After you answer **Y** to confirm the selection, the System Configure program will ask some additional questions, based on your specific hardware.

If you are configuring for an Apple II or II Plus, it asks whether you have installed the shift key modification:

```
HAVE YOU INSTALLED THE GAME I/O PORT
SHIFT KEY MODIFICATION (Y/N)?
```

This is a hardware modification to the original Apple II that allows the SHIFT key to function as on a regular typewriter. There is a detailed discussion of this hardware modification in Chapter 6. If you have installed this modification as described (using pin 4 of the GAME I/O socket) answer **Y** for Yes. Note: If you have installed an alternate type of shift key modification, such as the Videx Enhancer, you must run both the PIE CONFIGURE and FORMAT CONFIGURE programs (see Chapter 4) to inform PIE Writer of your modification.

If you are configuring for an Apple II or II Plus 40 column screen, the program asks if you have installed a lower case adapter:

```
HAVE YOU INSTALLED A 40 COLUMN
LOWER CASE ADAPTER (Y/N)?
```

The original Apple II cannot show lowercase letters on the 40-column screen, so unless you specify otherwise PIE Writer uses regular uppercase letters to represent uppercase letters. If you have installed a lowercase adapter, type **Y** (for yes) to display true upper and lower case.

If you have installed a Videx Videoterm 80-column board, the program asks:

```
HAVE YOU INSTALLED THE OPTIONAL VIDEX
INVERSE CHARACTER SET EPROM (Y/N)?
```

If you have installed the inverse character set EPROM, type **Y** so PIE Writer can display control characters using inverse video.

If you have an Apple //e, you are asked:

```
DO YOU WANT TO USE;

 1. THE APPLE //E ARROW KEYS FOR
    CURSOR MOVEMENT, OR

 2. THE STANDARD APPLE II PLUS
    KEYBOARD LAYOUT

WHICH ONE?
```

The Apple //e has four arrow keys located in the lower right-hand corner of the keyboard, which can be used for cursor movement. Unfortunately, the control codes generated by the hardware for these keys are not the same as those used for cursor movement by the Apple II Plus (and other) versions of PIE Writer. Therefore the keyboard layout had to be somewhat re-arranged to accomodate the use of the arrow keys, TAB key, and DELETE key on the //e. For instance, the cursor-up command in PIE Writer is normally CTRL-E (it is part of a "cursor diamond" formed by CTRL-S, CTRL-E, CTRL-F, and CTRL-C). But on the //e the cursor-up command may re-assigned to ↑ (which is the same as CTRL-K).

If you will be using PIE Writer only on the Apple //e, select **1** to obtain full use of the dedicated keys on your computer. However if you will be using PIE Writer on both an Apple //e and another computer, it is suggested that you select **2** to stay with the standard PIE Writer keyboard layout to make switching between machines easier.

Selecting one keyboard arrangement or the other also selects the appropriate version of the LESSON file, introduced in Chapter 2, and the PIE HELP file, which is available for reference from within the editor. When reading about text editor commands in Chapter 2 or Chapter 5, be sure to refer to the appropriate section (standard keyboard *vs* //e keyboard) based on the selection you have made.

The Franklin ACE fails to implement one Apple key sequence—CTRL-SHIFT-N. If you have a Franklin ACE, the program asks if you want to substitute the → key (CTRL-U) for the missing CTRL-SHIFT-N:

```
DO YOU WANT TO SUBSTITUTE CTRL-U
(-> KEY) FOR CTRL-SHIFT-N (Y/N)?
```

If you type **Y** this means you will lose the software SHIFT function, but will be able to do word delete using → instead.

The Franklin ACE has a "real" shift key and shift lock, so losing the software function is not critical. However you will not be able to enter four ASCII characters which are not present on either the Apple II or Franklin keyboards: | (vertical bar), ` (grave accent), \ (backslash), and rubout. If you need to use one of these characters, type **N** to retain the original → function, but you will lose the various forms of the CTRL-SHIFT-N function, including word delete. (You can still delete lines, however, using CTRL-K.)

After you have answered the appropriate questions, the program displays the message:

```
PRESS RETURN TO CONFIGURE THIS DISK
OR TYPE CTRL-C TO ABORT
```

If you are ready to configure this copy of the PIE Writer disk, press **RETURN** and PIE Writer will begin the configuration. This takes a few minutes, so be patient. When the configuration is complete, the System Menu (described in the next Chapter) is displayed. The PIE Writer disk is now configured to run with your system.

If you have discovered that you answered one of the above questions incorrectly, or you don't want to configure the disk for some other reason, type **CTRL-C** to abort the configuration process.

If you have a hard disk, you need to run both PIE CONFIGURE and FORMAT CONFIGURE to set the program disk volume (see Chapter 4). If you want to use special features of your printer, such as boldface or underline, you should run FORMAT CONFIGURE (Chapter 4) to inform the FORMAT text processor program of your particular printer type.

Notes About The System Configure Program

1. *You may type* **CTRL-C** *in response to any prompt, which will exit the System Configure program. You may immediately re-run the program by just typing RUN.*

2. *If there is no board installed in slot 3, the program assumes 40 column mode, and all output goes to the 40-column screen. If there is an 80-column board in slot 3, and its type is recognized by the program (using the Pascal 1.1 protocol), slot 3 is initialized (using PR#3) and output will go to the 80-column board. If there is a card in slot 3, but it is not recognized as an 80-column board, the program remains in 40-column mode, and then asks whether to configure for 40 or 80-column. If 80 column is selected, a list of board types is presented. After choosing one of the 80-column boards, output from the System Configure program will continue to be displayed on the 40-column screen until the configuration is complete.*

3. *The system type (Apple II Plus or Apple II Integer) is determined by whether the Applesoft or Integer version of the System Configure program was run (it assumes the disk was initially booted). It is possible to configure an Applesoft version of PIE Writer on a Apple II (Integer), or vice versa, but you must have both Applesoft and Integer BASIC available to do this (either by having a firmware card or language card in slot 0). If you have a language card, and the BASIC that is needed is not currently available, the program will ask you to exit, boot your DOS 3.3 System Master disk to load the necessary BASIC into the language card, and then re-run the System Configure program (by typing either RUN PIE WRITER: WORD PROCESSING, if you are configuring an Applesoft version, or RUN APPLESOFT, if you are configuring for an Apple II Integer machine).*

4. *If you are configuring an Apple II Integer version of PIE Writer to run on the language card, Applesoft BASIC must also be available to complete the configuration. If it is not currently loaded into the language card, the program will ask you to exit and boot your DOS 3.3 System Master disk as described above.*

2

The PIE Text Editor and Command Processor

Chapters 2 and 3 will teach you the fundamentals of using PIE Writer to create, edit or change, and print your documents. Both chapters assume that you have made a backup disk, and it has been configured.

PIE and FORMAT work together in the PIE Writer system. The PIE Text Editor, introduced in this chapter, allows you to create and change text for your letters, memos, books, or whatever. The PIE Command Processor, also introduced in this chapter, manages files that you create with the Text Editor. The FORMAT Text Processor, introduced in Chapter 3, allows you to layout and print a document according to your specifications.

The PIE Writer system diskette includes a lesson that teaches you all the Text Editor commands you will need to enter and edit your text on the screen. However, you can experiment with the Text Editor on your own if you want.

HOW TO USE THE TEXT EDITOR

When you boot the system diskette, the System Menu is displayed on the screen:

```
        SYSTEM MENU

1 - PIE TEXT EDITOR
2 - FORMAT TEXT PROCESSOR
3 - PIE CONFIGURE
4 - FORMAT CONFIGURE
5 - EXIT

     WHICH ONE ?
```

1. *Type* **1** *to respond to the "WHICH ONE?" prompt. The disk drive will whirr briefly, then the next screen of information will be displayed:*

```
FILENAME:       ?

LENGTH:         0
MEMORY LEFT:    22015

COMMAND?:
```

2. *To take the Text Editor lesson, type* **LOAD LESSON** *and press* **RETURN**. *If you mistype, press* **CTRL-X** *to cancel the line, then repeat this step. If you would rather use the Text Editor without taking the lesson, go directly to step 3 instead. If you have loaded LESSON, the next display is:*

```
FILENAME:       LESSON

LENGTH:           11252
MEMORY LEFT:      10763

COMMAND?:
```

3. *Type* **%H**, *then press* **RETURN**. *This option loads the PIE HELP file into memory. When the COMMAND?: prompt is displayed again, type* **E**, *then press* **RETURN.**

If you have loaded the lesson file, the first of several Text Editor lessons appears on your screen. First, read the introduction to the lesson in the text below. Note that each new Text Editor command is listed in bold with a brief description of its function. Then, follow the instructions on the screen for each lesson.

If you are working in the Text Editor with the lesson, practice the commands as they are introduced. Enter text anywhere on the screen. If you go beyond the right-hand margin and the window shifts to the right, press **RETURN** to get back to the main page.

To begin the whole lesson over, or to go back to where you began working with the Text Editor, type **CTRL-T**.

If you still need help remembering the function of certain commands, press **ESC H RETURN** to display all the simple Text Editor commands and their functions; press the **RETURN** key to return you to the point in the file where you left off.

PIE Command Key	Function
ESC H RETURN	Displays all simple Text Editor commands and their functions; press return to continue

IMPORTANT NOTE: This manual documents two sets of lessons. If you are using an Apple II, Apple II Plus, or Franklin ACE, then refer to the first set lessons for the Standard Keyboard.

If you are using an Apple //e *and* have selected option 1 in the System Configure program to use the Apple //e arrows keys for cursor movement, then refer to the second set of lessons for the Apple //e Keyboard. However if you selected option 2 in the System Configure program to use the Apple II Plus keyboard layout, then you want to refer to the first set of lessons for the Standard Keyboard.

PIE TEXT EDITOR LESSONS FOR THE APPLE //E KEYBOARD

Lesson 1: The Cursor

When the first Text Editor window appears on your screen, the cursor flashes in the top left-hand corner of the window.

When you want to scroll forward to the next page, use the **CTRL-V** (page down) command. The **CTRL-V** command advances the screen and cursor 21 lines, or 1 page, at a time.

PIE Command Key	Simple Function
CTRL-V	Advances screen 21 lines (1 page) at a time

Lesson 2: Home and Bottom Home

The second lesson describes the function of the **CTRL-G** (home) command. **CTRL-G** is a toggle command that moves the cursor to the top left-hand and the bottom left-hand corner of the screen.

Press **CTRL-G** to send the cursor Home from any cursor position on the screen. Press **CTRL-G** again to send the cursor to Bottom Home.

PIE Command Key	Simple Function
CTRL-G	Homes the cursor at the top left-hand or bottom left-hand corner of the screen (toggle)

Lesson 3: Simple Cursor Movement

The simplest cursor movements (up, down, left, and right one space) have been positioned on your keyboard so that you can perform them easily with one hand. At the bottom right corner of the Apple //e keyboard are four "arrow keys" that are used to move the cursor in the direction of each arrow. Each of the arrow keys also has an equivalent **CTRL** key sequence as shown in the table below.

Notice that when you move the cursor with these commands, the text the cursor passes over remains intact.

PIE Command Key	Simple Function
↑ or **CTRL-K**	Moves the cursor up 1 line
↓ or **CTRL-J**	Moves the cursor down 1 line
← or **CTRL-H**	Moves the cursor 1 space to the left
→ or **CTRL-U**	Moves the cursor 1 space to the right

Practice the cursor commands you have just learned. These commands are probably used more often than any other group of commands, so it is important to become familiar with them.

The Apple //e uses automatic repeat. As long as you hold any key down, the character will be automatically repeated. Use the automatic repeat function to make the cursor move up, down, left, or right faster.

Lesson 4: The Status Line

The Status Line is displayed just below the bottom border of the window on the screen. The Status Line displays information about the cursor position and the current text entry "mode."

The Status Line also keeps track of other details, such as INSERT MODE, and displays them on the screen. Keep an eye on the Status Line as you work.

Lesson 5: Beyond the Right-Hand Margin

The PIE Writer screen can display only 38 characters horizontally at a time {78 in the 80-column version}. There is additional typing room beyond the right-hand margin, though.

A right caret (>) on the right-hand side of the main page indicates that there is text beyond the right-hand margin on that line. Use the → key to move the cursor to the right until the right-hand side of the page appears.

As you type the 39th {or 79th} character in the line, you will hear the bell column tone. The bell column is the 38th {or 78th} column. It is displayed as a reverse video dash at the top right-hand corner of the main page window. This setting is a system default. You can set the bell tone in any column you want, or clear the bell column completely, with an advanced command (introduced in Chapter 5).

Press **RETURN** to get back to the main page.

Lesson 6: Tabs

You can move back and forth across the page quickly by using the tab commands, **TAB** or **CTRL-I** (tab right) and **CTRL-A** (tab left). Either the **TAB** key or **CTRL-I** moves the cursor one tab stop to the right; **CTRL-A** moves the cursor one tab stop to the left.

Each tab stop is indicated by a plus (+) sign at the top of the window. These settings are a system default. You can change the position and the number of tab stops with more advanced tab commands (introduced in Chapter 5).

If you tab the cursor beyond the right-hand margin, you can get back to the main page by tabbing left with **CTRL-A** or by pressing **RETURN**.

PIE Command Key	Simple Function
TAB or **CTRL-I**	Moves the cursor 1 tab stop to the right
CTRL-A	Moves the cursor 1 tab stop to the left

Lesson 7: Moving the Cursor to the Beginning or End of a Line

The **CTRL-B** command, like **CTRL-G**, toggles back and forth between two different functions. Press **CTRL-B** once to move the cursor to the end of the line, even if the end of the line is beyond the right-hand margin. Press **CTRL-B** again to move the cursor back to the left margin.

The **CTRL-B** command makes it easy to add text at the end of a line or begin a new sentence there.

PIE Command Key	Simple Function
CTRL-B	Moves the cursor to the beginning or end of a line (toggle)

Lesson 8: Scrolling Pages and Lines

You have been taught how to scroll the entire page forward using **CTRL-V** (page down). You also can scroll the page up or down a screen or a line at a time.

CTRL-R (page up) lets you scroll back a page at a time to review text.

You also can scroll up a single line at a time with **CTRL-Y** and down a line at a time with **CTRL-N**.

PIE Command Key	Simple Function
CTRL-V	Advances screen 21 lines (1 "page") at a time
CTRL-R	Reverses screen 21 lines (1 "page") at a time
CTRL-Y	Reverses screen and cursor 1 line at a time
CTRL-N	Advances screen and cursor 1 line at a time

Lesson 9: Entering Upper and Lowercase Characters

The **SHIFT** key on the Apple //e functions as on a normal typewriter, and allows you to enter both uppercase characters, and upper-register symbols (!, @, # etc.) and punctuation marks on the keyboard.

The Apple //e also has a **CAPS LOCK** key. When depressed (locked), all characters entered will be in uppercase. Press it again to unlock it.

Enter one or more letters in uppercase, in combination with lowercase letters. Type all over the screen. Practice using the **SHIFT** and **CAPS LOCK** keys to learn easy upper and lowercase character entry.

If you shift right after typing the 38th {or 78th} character in a line, press **RETURN** to get back to the main page.

Lesson 10: Destructive Cursor Moves

The first nine Text Editor lessons cover cursor control, window control, and text entry. Lessons 10 through 13 teach simple editing commands.

The **space** bar and **CTRL-S** (backspace) key each perform a simple cursor movement, but they also erase whatever text characters are in their path.

The **space** bar deletes a character, then moves the cursor 1 space to the right of where that character was.

The **CTRL-S** command deletes a character to the left as it backspaces.

Note that when deleting the characters on the screen, only the lesson in memory changes—but the lesson file is still stored on the disk intact for you to use again.

PIE Command Key	Simple Function
space	Deletes a character and moves the cursor 1 space to the right
CTRL-S	Deletes a character to the left as it backspaces

Lesson 11: Deleting a Character

Either the **DELETE** or **CTRL-D** key deletes a character or a blank and shifts the rest of a line left toward the cursor. Position the cursor on any letter you want to delete, and type **DELETE** or **CTRL-D** to edit a misspelled word.

PIE Command Key	Simple Function
DELETE or **CTRL-D**	Deletes a character at the cursor and moves the other characters on the line 1 space to the left

Lesson 12: Inserting a Character

The **CTRL-P** (insert) command allows you to enter characters anywhere within a word or a line.

CTRL-P toggles between entering and leaving the insert mode. Press **CTRL-P** to enter the insert mode, then enter as many characters in the middle of a word or a line as you want. All the characters on the line, beginning with the character at the cursor position when you first enter the insert mode, shift right as you enter text.

Press **CTRL-P** again to leave the insert mode.

PIE Command Key	Simple Function
CTRL-P	Begins or ends the insertion of characters at the cursor

Lesson 13: Shifting Text Left or Right

You can shift a line of text left or right while you are in the insert mode.

First, type **CTRL-P** to enter the insert mode.

Now, to shift a line to the right, press the **space** bar. All text on the line at or to the right of the cursor position before you entered the insert mode shifts right.

To shift a line to the left, press the **CTRL-S** (backspace) key. The **CTRL-S** key lets you "drag" a line to the left.

The editing commands taught in these last few lessons should make one thing clear: You need not enter text perfectly the first time with PIE Writer. You can always review and edit your document later.

PIE Command Key	Simple Function
CTRL-P space	Shifts a line of text to the right 1 space
CTRL-P CTRL-S	Shifts a line of text to the left 1 space

Lesson 14: Returning to the Beginning of a File; Leaving the Text Editor and Entering the Command Processor

Type **CTRL-T** to return to the first page of your file. **CTRL-T** takes you to the beginning of any file created with the Text Editor.

Remember that you can scroll backward or forward a page or screen at a time. To go back to a specific place in your file, use **CTRL-R** (page up), **CTRL-V** (page down), **CTRL-Y** (scroll up), and **CTRL-N** (scroll down).

When you have finished working with the Text Editor, type **CTRL-@** to leave the Text Editor and enter the Command Processor. (**CTRL-@** is the same as typing **CTRL-2** on the //e.)

PIE Command Key	Simple Function
CTRL-T	Moves the cursor to the beginning of a file
CTRL-@	Leave the Text Editor and enter the Command Processor

The manual documents more sophisticated and powerful word processing commands that can be used by any PIE Writer user. All these commands are introduced in Chapter 5.

When you leave the Text Editor with **CTRL-@** you enter the Command Processor.

THE COMMAND PROCESSOR

The Command Processor (CP) manages all the operating files on the system diskette and all the document files you create using the Text Editor. Some of the functions of the CP are:

1. *Loading a file from the diskette so you can review, edit or add to it.*
2. *Saving text you have written on a diskette file and labeling the file with a name you specify.*
3. *Clearing the memory so that you can begin a brand new file.*
4. *Reminding you of the filename of the text you are working on.*
5. *Reporting on the number of words or lines in a file.*

6. *Transferring you back and forth between the CP, the System Menu, and the Text Editor.*
7. *Sending text you have written to be printed using the FORMAT Text Processor.*

Again, these are only a few of the Command Processor's capabilities.

Before you learn any commands, there is one more concept with which you should become familiar: the "text buffer."

Saving the Text Buffer

The text buffer can be thought of as the computer's short-term memory. It sets aside a certain amount of space for you to enter a document and remembers all the text that you enter. But it remembers your file only while you are working on it. To store the text that is in buffer for future reference, you must "save" it on a diskette.

IMPORTANT NOTE: Everything in the buffer will be erased completely if any one of the following happens:

- *You turn the computer off or the power goes off.*
- *You load another file from the diskette for review or editing. The file from the diskette replaces the text that was in the text buffer.*
- *You tell the CP to begin a new file. The text buffer will clear out any text it is holding and stand by to receive new text.*

Before loading another file, beginning a new file, or whenever you have typed anything you cannot afford to lose, ALWAYS make sure you save the contents of the text buffer. Get in the habit of saving often to save yourself many hours of grief.

You will learn how to save a file later. First, you will learn how to begin a new file.

Beginning a New File

Every time you want to prepare a new document, you must begin a new file for it. To begin a new file, with the COMMAND?: prompt displayed on the screen:

```
COMMAND?:
```

1. *Type* **N** *(or* **NEW***)*
2. *Press* **RETURN**

The **N** you type tells the CP to clear out the text buffer and prepare to receive new text; **RETURN** signals that you have finished entering your command and are ready to have it carried out. The CP requires a **RETURN** after every command before it can perform the required function.

Before clearing the buffer, the CP will give you one last chance to save any text in the text buffer, prompting as shown below on the left. Since you have not yet typed anything into the buffer yet, it is safe to give the go-ahead, shown on the right:

```
OK TO CLEAR?:
```

1. *Type* **Y** *(for Yes)*
2. *Press* **RETURN**

The text buffer window appears. This is the first blank page onto which you may begin entering text.

You have begun a new file, leaving the Command Processor and entering the Text Editor for the moment.

CP Command Review	Function
N (or NEW) RETURN	Clears the text buffer, and opens a new file

Typing Modes

When you enter the Text Editor for the first time, you can begin entering text in the system's default Manual mode.

To enter text smoothly in Manual mode, you need make no adjustments, just follow the directions below:

- *When you get to the end of a line, press* **RETURN**.
- *If you type onto the right-hand side of the page, press* **RETURN** *to get back to the main page.*

After you have typed long enough to get a feel for the restrictions of Manual mode (pressing **RETURN** at the end of each line), you can switch to PPWRAP mode for easier text entry. To change the system default and use the PPWRAP mode:

1. *Press* **ESC**
2. *Press* **RETURN**

Note that when you enter PPWRAP mode, the Status Line says "PPWRAP."

In PPWRAP mode, your typing speed should almost double, because you do not need to press **RETURN** when the cursor reaches the end of a line. The lines "wrap" around the screen automatically.

PIE Writer is now set for you to start typing streams of sentences in PPWRAP mode without having to press **RETURN**.

PIE Command Key		Function
ESC	RETURN	Sets PPWRAP typing mode

When you enter the PPWRAP mode, you can tab to the beginning of words to the left (**CTRL-A**) or right (**CTRL-G** {Apple //e: **TAB** or **CTRL-I**}) of the cursor position. Notice that the plus (+) signs disappear from the top of the screen.

Again, there are many additional cursor, screen control, and editing commands available in the Text Editor. These commands, providing such features as word tabbing, searching and replacing, and line moving are covered in Chapter 5.

Now, though, you will learn how to get back to the Command Processor and save a file. When you have practiced enough typing in the PPWRAP mode, read on.

Saving a File

Right now, everything you have typed exists only in the text buffer. To place your text in the computer's long-term memory, you must save a copy of it on the diskette.

The first step in saving a file is to leave the Text Editor and enter the Command Processor with **CTRL-SHIFT-P** {Apple //e and Franklin: **CTRL-@**} command.

After you leave the Text Editor, your text will disappear and you will be prompted by the CP:

```
COMMAND?:
```

To save the file:

1. *Type* **S** *(or* **SAVE***).*
2. *Type at least one blank space.*
3. *Type a filename of your choice. (For users with 2 disk drives, you may type* **,D2** *after the filename.) Filenames must begin with a letter; they cannot contain any commas; and they must be no more than 30 characters long, including blank spaces. If you do not begin your filename with a letter, or if your filename contains a comma, you will receive a PIE command error message:*

```
*PIE  CMD  ERR
```

If you receive the message at left, return to step 1.

If you type a name longer than 30 characters, the CP will use only the first 30

characters as your filenmame. If you mistype your filename, type **CTRL-X** *to cancel the line and retype it correctly. The CP cannot edit text with the Text Editor commands.*

4. *Press* **RETURN**.

After the disk drive whirrs for a minute, the CP will save your file, report the following information, and await your next command:

```
FILENAME:

LENGTH:
MEMORY LEFT:

COMMAND?:
```

The number reported next to LENGTH: tells how many characters your file has used. MEMORY LEFT: tells you how many more characters it would take for your file to fill up the text buffer.

CP Command Review	Function
S (or **SAVE**) **filename** **RETURN**	Saves, or writes, a file to diskette as "filename"

A Caution About File Lengths

If you are typing a particularly long file, the text buffer may approach its capacity. When this happens, you will be notified with a warning beep, and the Status Line will display the message "ALMOST OUT OF MEMORY." Each additional character you type will produce another beep. Stop typing at the next convenient point, such as at the end of the paragraph, and save your file on the diskette. You can then continue your document in a new file under another, closely related, filename, such as ANYFILE PART 2.

It is EXTREMELY important that you do not type more than another 21 lines. Doing so will cause the buffer to run completely out of memory space. Your typing will overwrite the disk operating system, and you will not be able to save any of your file.

Again, when you get the "ALMOST OUT OF MEMORY" warning:

1. *Stop typing as soon as conveniently possible.*
2. *Save the file.*
3. *Continue your document in a new file with a different, but appropriate, name.*

A word of reassurance: You are unlikely to type on forgetfully after the first warning, since the beep will sound with each additional character you type.

Editing a File

Now that you have saved your file, you have two copies of the text: one on the diskette, and one still in the text buffer. To edit your text, you must return to the on-screen PIE Text Editor.

Getting the file back on the screen from the text buffer is practically instantaneous. After saving your file, the file status is redisplayed as shown below at left. Respond to the prompt with the commands on the right.

```
FILENAME:      ANYFILE

LENGTH:
MEMORY LEFT:

COMMAND?:
```

1. *Type* **E** *(or* **EDIT***)*
2. *Press* **RETURN**

Your text is back on the screen.

CP Command Review	Function
E (or **EDIT**) **RETURN**	Returns you to the Text Editor to review current file for editing

Now make some changes or additions to your text that is already on the screen. If you have loaded the HELP file using the **%H** at the Command Processor level, you need only press **ESC H RETURN** to display all the simple commands and their functions if you need help remembering any editing or scrolling commands. Press **RETURN** to get back to the point in the file where you left off.

Here is an important concept to grasp: If you make changes in a file, you must save the file again. Otherwise, the changes will not be recorded and will disappear the first time you clear the contents of the text buffer.

When you are ready to save your changes, try the next exercise.

Displaying Your Filename

When you save an edited file that contains changes of any sort, you may want to save the changed file using the same name you gave the original file. Otherwise, you will be

saving two files: one with changes and one without.

Whenever you return to the CP from the Text Editor, your filename is displayed. You can also ask the CP to display the current filename by typing **? RETURN** in response to the COMMAND?: prompt.

Now, repeat the same steps you performed the first time you saved the file (**S filename RETURN**). The edited version of your text replaces the earlier version of the file on the diskette. The next time you load this file from the diskette, all your editing changes will be incorporated.

CP Command Review	Function
? RETURN	Displays current filename

Viewing the Catalog of Files

Prove to yourself that the file has been saved on the diskette. Clear the text buffer by turning the computer off, then on again. (If your diskette does not boot automatically, see "Getting Started" in Chapter 1.) After the diskette boots, the System Menu will appear. Press **1** to enter the PIE Command Processor.

You can get the CP to provide you with a list of all the files stored on a diskette. Each diskette contains a "catalog," or index, of all its files. This catalog includes both the files that enable the PIE Writer system programs to run as well as your document files.

To get the CP to display the catalog on the screen, respond this way to the prompt:

1. *Type* **C** *(or* **CATALOG***)*
2. *Press* **RETURN**

If you have more than one disk drive, type **C,D1** or **C,D2** to catalog the desired diskette.

The list of files appears to the right of the file type abbreviations and file lengths. It should be easy to find the name of your file on the list.

CP Command Review	Function
C RETURN	Catalogs the current diskette; type **C,D1** or **C,D2** instead of **C** if you have more than 1 disk drive

Loading a File

Load your file (if you have not already), in response to the CP prompt, to see if the

edited version was remembered by PIE Writer.

1. *Type* **L** *(or* **LOAD***), followed by one blank space and the name of your file (typed EXACTLY as listed in the catalog)*
2. *Press* **RETURN**

The name of your file reappears on the screen.

If you mistype your filename, you will be informed that the file is not on the diskette, with the computer's response:

```
FILE NOT FOUND
```

If this happens, simply retype the filename after the next COMMAND?: prompt. The CP is obedient, but literal. You must always be precise.

CP Command Review	Function
L filename RETURN or **LOAD filename RETURN**	Loads a file into the text buffer

After loading the file, you can re-enter the Text Editor again to edit the file by typing **E RETURN**.

PIE Summary

In Chapter 2, you have learned:

1. *Basic cursor movement, window control, and most simple Text Editor commands.*
2. *How to begin a new file in the Command Processor*
3. *How to save a file.*
4. *How to edit a file in the text buffer.*
5. *How to display a filename.*
6. *How to display a catalog of the files on a diskette.*
7. *How to load a file from the diskette.*

All the basic PIE commands already introduced and all the advanced PIE commands are fully documented in Chapter 5, "Reference Section." Also, all the PIE commands are noted in an abbreviated form on the PIE Writer Reference Card.

Now that you know how to create a perfect document with PIE, the next step is to learn how to print it. The FORMAT Text Processor controls the layout and printing of the documents you create with the PIE Text Editor and Command Processor.

3

The FORMAT Text Processor

The FORMAT Text Processor is the third part of the PIE Writer Word Processing System. It controls the layout, design, and printing of your document.

There are two parts to FORMAT. The first part lets you control the layout and design of your document with special commands you'll enter right along with your text while you are working with the Text Editor. The second part of FORMAT lets you specify printing options, such as which document you want to print, the number of copies you want printed, and whether you want to print all or only certain pages.

In this chapter, you will learn how to use FORMAT dot commands to specify the appearance of a document and FORMAT Text Processor commands to print it out.

LESSON 1: INTRODUCTION

To specify the layout or design of your document, you enter FORMAT dot commands with the rest of your text while working on the PIE Text Editor screen. The instructions appear on the screen as you type them. Later, you instruct the FORMAT Text Processor to execute them.

Entering FORMAT Dot Commands

When entering FORMAT dot commands while working on the PIE screen, every command must:

- *begin with a dot*
- *begin at the left margin*

- *be sole occupant of an entire line*
- *precede the text it applies to*

Commands may be entered in either uppercase or lowercase.

For practice in entering FORMAT commands, load the sample file named COLUMNS, included on your PIE Writer diskette. (If you are currently practicing on the PIE screen, exit to the CP with a **CTRL-SHIFT-P** {Apple //e and Franklin: **Ctrl-@**}.) The CP will prompt:

```
COMMAND?:
```
Load the practice file by typing **L COLUMNS**

```
COMMAND?:
```
To see the COLUMNS file on the PIE screen, type **E**

When you have COLUMNS on the screen:

1. *Position the cursor on the "A" in "ADJECTIVES"*
2. *Press* **CTRL-I** {*Apple //e: press* **Ctrl-E**} *to insert a blank line (you will learn more about this command in Chapter 5)*
3. *Type* **.NF** *at the beginning of the new blank line*
4. *Position the cursor on the "S" in "STRONG"*
5. *Press* **CTRL-I** {*Apple //e: press* **Ctrl-E**}
6. *Type* **.FI** *at the beginning of the line*

When you are done, your screen should look like this:

```
.NF                                          .nf initiates the "no-fill"
ADJECTIVES      NOUNS       VERBS                    mode

WINSOME         DOGS        EAT
GIANT           BONES       GROW
.FI                                          .fi begins the "fill" mode
STRONG          DOUBTS      CRY
SOFT            OATHS       FLOAT
SMALL           LOANS       AFFECT
SUBTLE          PLANS       SHIFT
```

Displaying a File

To see exactly how the **.fi** and **.nf** commands affect your text, you should review the "formatted" file.

Leave the Text Editor with a **CTRL-SHIFT-P** {Apple //e and Franklin: **Ctrl-@**}. You will be prompted by the CP:

```
COMMAND?:
```

To summon the FORMAT Text Processor, type **F**

If you have a 48K version of PIE Writer, the disk will come on for a moment. When the drive stops whirring, the Text Processor's first prompt appears on the screen:

```
OUTPUT TO PRINTER...
```

For this exercise, press **RETURN**

The Text Processor is asking you for a command. It wants to know where to send, or "output," the formatted text you are about to identify. You can output your file to a printer, a diskette, or to the screen. For this exercise, you selected the screen by pressing **RETURN**.

Another prompt follows:

```
PAGE LIMITS...
```

To output the whole file, press **RETURN**

FORMAT can output either certain pages of your document or the entire file. Here, the Text Processor asks which pages of the file you want it to work on. By pressing **RETURN**, you selected the whole file.

```
INPUT FROM DISK (D) OR
PIE MEMORY BUFFER (M)?
```

To input a file in the PIE memory buffer, type **M**

FORMAT wants to know where to find the "input" file. Text you have just entered or loaded is stored in the memory buffer. For this exercise, the file COLUMNS, to which you just added FORMAT dot commands, is still in the memory buffer.

FORMAT can also find and load a file that is on the diskette. If your file was one already stored on diskette, you would have typed **D** in response to the last prompt. FORMAT then would ask you for:

```
FILENAME?
```

To get a file from diskette, type the **filename** *and press* **RETURN**

Before FORMAT can get a file from the diskette, you have to enter the name of the file you want loaded. (Here, since you typed **M** instead of **D**, FORMAT goes straight to work on COLUMNS, which is stored in the memory buffer.)

Remember that the commands you entered in the COLUMNS file are not yet saved on the diskette, but they are still in the memory buffer.

After you have told FORMAT where to find your text and, if on the diskette, what its name is, the cursor will flash at the bottom left-hand corner of your screen, waiting for your next command. Tap the **space** bar once.

When you tap the **space** bar, formatted text from the COLUMNS file begins scrolling from the bottom of the screen to the top very quickly. You can tap the **space** bar to stop the scrolling, then tap it again when you are ready to advance the text.

The screen will stop scrolling automatically after displaying 24 lines of formatted text. It also stops at the end of every formatted page (you will learn how to specify the length of your pages with a dot command later), and displays a series of dashes, just above where the cursor flashed. Again, tap the **space** bar to advance the text.

When the entire file has been displayed on the screen, the flashing cursor again appears in the bottom left-hand corner.

Reading Formatted Text on the Screen

For users with the 40-column version of PIE Writer, the formatted text scrolls up the screen as shown below when you tap the **space** bar:

```
ADJECTIVES      NOUNS      VERBS

WINSOME         DOGS       EAT
GIANT           BONES      GROW
STRONG DOUBTS CRY SOFT OATHS FLOAT SMALL
 LOANS AFFECT SUBTLE
PLANS SHIFT
```

Note the effects of the **.nf** and **.fi** commands on the text in the COLUMNS file.

The "fill" mode is a FORMAT default, so formatted text is always displayed in fill mode when it is sent to the screen, printer, or diskette, unless otherwise instructed with the **.nf** command. In the fill mode, FORMAT normally gathers enough characters (and spaces) to fill a 65-character line of output text, without breaking words in two. When a file is printed in fill mode, each line contains no more than 65 characters, regardless of the number of characters each line on the screen contains.

The **.nf** command you inserted at the beginning of the file instructs FORMAT to stop filling, initiating the "no-fill" mode. No-fill lets you enter tabular material and output it as it appears on the screen. This allows the first three lines of COLUMNS to remain within actual columns.

The **.fi** command reinstates the fill mode, forcing the last four lines of column material to run together as if they were part of a paragraph.

The formatted line beginning with "LOANS" is appended to the line ending with "SMALL" in the 80-column version of PIE Writer. The line breaks after "SMALL" in the 40-column version of PIE Writer because that version can display only 40 characters per line on the screen. When the text is printed, though, both the 40- and 80-column versions print the words from "STRONG" to "SUBTLE" on a single line.

Other FORMAT Commands

There are several other simple FORMAT commands covered in this chapter. These and more sophisticated FORMAT commands are covered in detail in Chapter 5 and are listed in abbreviated form on the Reference Card.

If no command is entered to alter one of FORMAT'S "permanent" initial values, then those values go into effect automatically. (Instructions on how to alter "permanent" FORMAT values are in Chapter 4, "PIE and FORMAT Configure.")

Most FORMAT commands take an 'n' value, however, allowing you to alter the initial "default" values temporarily. For instance, the **.ll** command allows you to alter the 65-character default value for line length initially set up by FORMAT. Entering an 'n' value immediately following the **.ll** command lets you make your printed document 'n' characters wide.

Finally, some FORMAT commands force the text that follows them to "break" off and begin a new line of output text, even when they are entered in the fill mode.

It is important to keep these concepts in mind if you decide to alter the "permanent" FORMAT values, which are now set up for typical file formatting. Changing values for certain parameters can make formatting a specific type of file much simpler.

LESSON 2: PRINTING A FILE

Transferring Between the Text Processor, the Text Editor, and the System Menu

To practice commands or enter text on the Text Editor and print them out with the Text Processor, you should be able to transfer back and forth between the two easily.

To leave the Text Processor and return to the Text Editor, press the **space** bar in response to the flashing cursor that appears after the COLUMNS file is displayed on the screen until the FORMAT Text Processor prompts:

```
RETURN TO PIE (Y/N)?
```
To return to the CP, type **Y**

To summon FORMAT quickly from the CP mode, you must respond to the CP prompt:

```
COMMAND?:
```
To summon FORMAT again, type **F**

You can transfer to PIE when you are being prompted for input and output information by the Text Processor also. When any of the FORMAT prompts appear, respond with an **ESC**ape. For instance:

```
OUTPUT TO PRINTER...
```
To summon the CP from FORMAT, press **ESC**

Again, you are asked whether or not you want to:

```
RETURN TO PIE (Y/N)?
```
To return to the CP, type **Y**

If you would rather rerun FORMAT, type N or press RETURN in response to the above prompt. The prompt below will appear. Type Y or press RETURN to rerun FORMAT.

```
RE-RUN FORMAT (Y/N)?
```
To rerun FORMAT, type **Y** *or press* **RETURN**

Typing **N** in response to both the RETURN TO PIE and RE-RUN FORMAT prompts produces an endless loop of the same prompts over and over again until you either:

1. *Type* **Y** *in response to one of the prompts*
2. *Type* **%M** *in response to either prompt, which returns you to the System Menu*
3. *Type* **%Q** *in response to either prompt, which transfers you to BASIC*

Printing Hard Copy

You can send text to your printer, allowing you to compare "hard," or printed, copies.

A formatted copy is printed using FORMAT, with all the commands executed. A literal copy, showing text and commands exactly as you entered them, is printed using the Command Processor.

To print a formatted copy of COLUMNS, with FORMAT commands executed:

1. *Make sure the printer is on and "on-line"*
2. *Summon the CP*
3. *Load COLUMNS, if it is not in the text buffer already*
4. *Type* **F** *and press* **RETURN**
5. *Respond to the FORMAT prompts as follows:*

```
OUTPUT TO PRINTER...          Type P

PAGE LIMITS . . .             Press RETURN

INPUT FROM DISK . . .         Type M
```

The printer should start printing, producing formatted copy as shown below:

```
ADJECTIVES      NOUNS      VERBS

winsome         dogs       eat
giant           bones      grow
strong doubts cry soft oaths float small
 loans affect subtle
plans shift
```

If your printer does not begin printing, check all your hardware connections and run through the five steps above once more. If your printer still does not work, you may need to reconfigure your system (see Chapter 4, "PIE and FORMAT Configure" for a complete explanation).

To print a literal copy of COLUMNS, showing text and FORMAT editing commands exactly as you entered them:

1. *Make sure the printer is on and on-line*
2. *Summon the CP*
3. *Load COLUMNS, if it is not in the text buffer already*
4. *Respond to the COMMAND?: prompt by typing* **>#1**
5. *Press* **RETURN**

If your printer card is not in slot #1, enter the card's correct slot number in place of the "1" in step 4.

The printer should produce a literal copy of COLUMNS, as shown below:

```
.NF
ADJECTIVES      NOUNS       VERBS

winsome         dogs        eat
giant           bones       grow
.FI
strong          doubts      cry
soft            oaths       float
small           loans       affect
subtle          plans       shift
```

LESSON 3: PARAGRAPH CONTROL

Standard Paragraphs

The command for a standard paragraph is one of FORMAT's most useful features. The **.pp** command signals the beginning of a paragraph. It also activates four other built-in commands, outlined below:

```
.PP
BEGINS A PARAGRAPH
```

.sp—*Space down 1 line.* Here, a .sp command leaves 'n' blank lines between the last line of text and the first line of the new paragraph. With the .pp set, there is 1 blank line between paragraphs (n = 1).

.ti—*Temporary indent +5.* A .ti is used to indent the first line of each paragraph 'n' spaces. The .pp command sets the temporary indent at +5 spaces (n = +5).

.ne—*Need 2 lines to bottom.* A .ne measures the space remaining on the current page as it has been set up for output. If there is not enough room for at least 'n' more lines of text, the new paragraph will begin on the next page. The .pp command sets n = 2.

.fi—*Fill.* The .pp turns on the fill mode.

Every time you insert a **.pp** command at the beginning of a paragraph, FORMAT automatically executes the other four commands.

The no-fill paragraph command, **.np**, begins a no-fill paragraph, which allows you to print text exactly as it appears on the screen. It does not set a temporary indent, but it does activate three other built-in commands, two of them the same as those affected above:

1. **.sp** *spaces down 1 line (***.sp 1***)*
2. **.ne** *has a "need" value of 2 (***.ne 2***)*
3. **.nf** *stops filling of text*

The **.np** allows you to alter the need value by entering your own value, paragraph by paragraph. For example, **.np 5** sets the need value at 5.

Other Paragraph Commands

Additional paragraph control commands override values set by **.pp** or **.np**, as described below:

.ps—*Modify paragraph spacing.* This command overrides .sp 1 to increase or eliminate space between paragraphs.

.pn—*Lines needed to begin paragraph.* A .pn overrides the default .ne 2 to alter the number of lines needed to begin a paragraph at the bottom of the page. This is helpful in assuring that there is room for 'n' lines of a paragraph beginning at the bottom of a page.

.pi—*Paragraph indent.* This overrides the default .ti +5 to alter the first line indent of each paragraph.

These three paragraph commands are activated by **.pp** or **.np**. Since a no-fill paragraph has no indentatiions, **.np** activates only **.ps** and **.pn**. A **.pp** will activate all three commands. Once entered, **.ps**, **.pn**, and **.pi** work every time you enter either the **.pp** or **.np** command to begin a paragraph.

For example, to set up:

- *3 blank lines between paragraphs*
- *an 8-space indent on the first line of paragraphs*
- *need room for 4 lines at end of page*

You enter:

```
.PS 3                          3 blank lines
.PI 8                          8-space indent
.PN 4                          Need 4 lines
.PP                            The .pp must be
THESE ARE THE COMMANDS           entered last
YOU SHOULD HAVE ENTERED.
```

If you enter another **.pp** to start a new paragraph, it will be formatted just like the first one. To check, print the paragraphs using FORMAT.

Once you have entered **.ps**, **.pn**, and **.pi** values, they are executed every time you begin a new paragraph. To return to the original default settings, load your practice file, then enter:

```
.PS
.PN
.PI
```

If you enter a **.pp**, type another paragraph, then print it, the return to the default values should be apparent.

One-Time Paragraph Indent

One final note about the **.pp** command: you can specify the first-line indent of a single paragraph by entering a 'n' value after the **.pp**. Otherwise, it works just like a plain **.pp**, activating **.sp**, **.ne**, and **.fi** commands.

For instance:

- **.PP**— *indents a single paragraph 5 spaces*
- **.PP 0**—*no indent*
- **.PP 10**— *indents a single paragraph to begin in 11th column*

FORMAT Command Review	Function
.pp n	Begin paragraph with optional first-line indent of 'n'
.ps n	Modify paragraph spacing
.pn n	Lines needed to begin a paragraph at the bottom of a page
.pi n	Set paragraph indent
.np n	Begin no-fill paragraph with optional one-time need value of "n" lines

The next lesson shows you how to use the paragraph commands, along with other commands, to format a letter.

LESSON 4: DESIGNING A LETTER

If you have been composing and printing your letters on a typewriter, you may be used to doing page layouts by eye. When producing letters or any business report with FORMAT, you should get used to specifying page layout by calculation and command. Now you will learn how to design a letter with normal dimensions:

- *the paper is 8.5 inches wide*
- *the print size is 10 characters per inch*

If your paper and print sizes are different, substitute your own measurements in the following calculations.

Calculating Margin Widths

On a centered page with equal left and right margins, each margin will be:

$$\frac{\text{page width - line length}}{2}$$

You know that the page is 8.5 inches wide. With FORMAT, it is easier to think in terms of character spaces. To convert inches to character spaces, just multiply the width of the page in inches by the characters per inch. In this exercise:

page width = 8.5 (inches) x 10 (characters per inch)
= 85 character page width

Next, you will need to specify the line length. For now, use FORMAT's standard line length. The standard line is 65 character spaces long.

Now that you know the page width and line length, you can figure out the margins:

$$\frac{\text{85 (page width) - 65 (line length)}}{2}$$

Each margin will be 10 spaces, or 1 inch, wide.

Setting Up the Left Margin

The **.po** (page offset) command helps you set up the left margin of your page. However, the space you allot for your left margin with page offset is not necessarily equal to the width of the left margin, for most printers.

Since the **.po** default is 0, you might suppose that, by default, the printer should begin printing at the left edge of the page. But, in fact, print begins only as far left on the page as the printer can manage. The area in which it cannot print becomes part of the left margin.

To determine the overall distance in which the printer cannot operate, print out any paragraph, like the paragraph below, preceded by a **.po 0** command, then measure from the left edge of the paper to the beginning of the print.

```
.PO 0
PAGE OFFSET IS SET AT 0.
THE PRINTER MARGIN IS THE
SPACE TO THE LEFT OF THESE
LINES.
```

Suppose that the printer leaves a half-inch margin before it begins to print. Half an inch equals 5 character spaces (print size is 10 characters to the inch). The left margin needs 10 spaces. The printer margin donates 5 of them; the **.po** command's 'n' value is used to make up the difference.

10 (left margin) - 5 (printer margin) = 5 (**.po** '5')

For a similar printer, the page offset 'n' would be 5 (or 10 minus the printer's margin width).

FORMAT Command Review	Function
.po n	Sets page offset

Line Length

The **.ll** command controls the number of character spaces per line, including indentations and blank spaces between words.

For this exercise, use the PIE Writer system default of 65 characters.

Right Margin

The right margin is the space left over when you subtract the left margin and the line length from the total page width.

10 (left margin) + 65 (line length) = 75
85 (page width) – 75 = 10 (right margin)

So, to set the margins for a centered page with standard 65-space lines:

```
.PO 5
TEXT FOLLOWING HAS
65-CHARACTER LINES AND
1-INCH LEFT AND RIGHT
MARGINS.
```

The **.po** *command is all you need to enter*

Work on setting up pages of different text widths and different margins until you fully understand how they are set up with FORMAT.

More layout practice follows. Remember the page offset value appropriate for your printer.

The Heading and Date

The lines between the top of the page and the heading, of course, vary with the length of the entire letter so that the letter is centered vertically on the page. For this exercise, assume that 10 lines of space will be needed before the heading. If the letter turns out to be shorter or longer than expected, the space above the heading can be altered after reviewing the formatted letter before it is printed.

Assuming that 10 lines of space are appropriate for this letter, you enter:

```
.PO 5                          Set the page offset
.SP 10                         Space down 10 lines
.NF                            Cancel the fill mode
.IN 33                         Indent the heading and
UNIVERSAL MACHINES, INC.         date to start at the
1000 MAIN STREET                 middle of the page
PARIS, TEXAS  52500              (33 characters)
.SP                            Allow 1 line space to date
JANUARY 1, 1999
.IN                            Cancel indent
```

What you have done is set the page offset for your printer, spaced down 10 blank lines, then indented the heading and date to the middle of the page (33 is half the line length), allowing 1 line space between them.

The **.in** command has the following functions:

.in—*Indent.* The .in command indents all the lines that follow it 'n' character spaces to the right of the page offset (.po) value. The .in command works in both fill and no-fill modes.

A **.nf** command is entered on the third line in the screen above so that FORMAT outputs each line exactly as it is entered. Otherwise, FORMAT would gather enough characters to fill up a 65-character line, then begin the next line at the 33-character indent.

The **.in** command at the end of the text above allows the next line entered to begin flush at the left margin because it returns the indent to the default of zero.

FORMAT Command Review	Function
.sp n	Insert 'n' blank lines
.fi	Set fill mode
.nf	Set no-fill mode
.in n	Indent subsequent text 'n' characters

Inside Address and Salutation

Now space down 4 lines and type the inside address, skip a line, and type the salutation:

```
.SP 4                              Space down 4 lines
UNIVERSAL PARTS, INC.              Type inside address
9999 FOREST AVENUE
ROME, NEW YORK  99999
.SP                                Allow 1 line space to
DEAR LADIES OR GENTLEMEN:            salutation
```

Since you are still in the no-fill mode (**.nf**) and the indent (**.in**) has been returned to its default value, the inside address will be set line by line, each line flush at the left margin.

If you had not cancelled the PIE Writer system's default fill mode, the heading, date, inside address, and salutation, could be broken line by line with a **.br** command.

.br—*Break in filling.* In fill mode, the .br command tells FORMAT to stop filling and make the next line a new one.

```
.SP 4                              Note that the .br command
UNIVERSAL PARTS, INC.                .instructs FORMAT to
.BR                                  begin a new line of text,
9999 FOREST AVENUE                   whereas the .sp command
.BR                                  begins a new line of text
ROME, NEW YORK  99999                AND inserts blank lines
.SP
DEAR LADIES OR GENTLEMEN:
```

Line breaks are built into many FORMAT commands. For example, when you enter a **.sp** the line following the command breaks to begin a new line, rather than fill in at the end of the line above the **.sp** command.

FORMAT Command Review	Function
.br	Break in filling

Body of Letter

Return to the fill mode before entering any text so that the body of your letter will be printed in 65-character lines. The **.pp** command returns you to the fill mode and skips one before beginning the first paragraph:

```
.PP                              Begin new paragraph
ENTER A PARAGRAPH HERE.
.PP                              Begin new paragraph
ANOTHER PARAGRAPH.
```

Now start typing paragraphs. Use the **.pp** command to insert a blank space between paragraphs, to break for a new line, and to indent the first line of the paragraph 5 spaces.

To make text entry easier, you can enter the PPWrap mode (see Chapter 2 for a review). Press **ESC** then **RETURN** for uninterrupted text entry.

The right-hand margin will not be justified automatically, but you can initiate the right justification of text with a **.ad** command:

.ad—*Adjust right.* This command increases the space between words to ensure that each line of text following the command is the same length when your document is printed. If your printer is capable of incremental spacing, and if you have specified one of the daisy wheel printers as the printer type in FORMAT Configure, the .ad command initiates incremental spacing as well as right justification (see Chapter 4 for a full explanation of this reconfiguration procedure).

.na—*No adjust right.* This command cancels right-margin justification and incremental spacing.

FORMAT Command Review	Function
.ad	Adjust right
.na	No adjust right

Closing and Signature

Skip a line, then enter the closing and signature on the same indent as you did the heading and date. To do this, and to make sure that both closing and signature appear on the same page, use the **.np** command:

```
.NP 6                            Allow 1 line space
.IN 33                           Set indent
YOURS TRULY,
.SP4                             Allow 4 lines of space to signature
YOUR NAME
```

Note that the **.np** command returns you to the no-fill mode, in which lines are printed out as typed, each one starting after the 33-character indent.

Saving and Reviewing the Letter

Now leave the Text Editor with a **CTRL-SHIFT-P** {Apple IIe and Franklin: type **Ctrl-@**}, save the file containing your letter, then transfer to the Text Processor:

`COMMAND?:`	*To save your letter, type* **S FILENAME**
`COMMAND?:`	*To transfer to the FORMAT Text Processor, type* **F** *then press* **RETURN**

To see your formatted letter, you will want to output FILENAME to the screen:

`OUTPUT TO PRINTER...`	*To output FILENAME to the screen, press* **RETURN**
`PAGE LIMITS...`	*To output the entire letter, press* **RETURN**
`INPUT FROM DISK...`	*Since you have saved the letter under FILENAME you can input from the buffer by typing* **M** *or you can input from the diskette by typing* **D.**
`FILENAME:`	*If you input from the diskette, type your FILENAME*

Tap the **space** bar to see your letter.

Notice that the indented parts of your letter, the heading, date, closing and signature, are indented 33 character spaces on the formatted screen. {Once the screen prints the 40th character on the line, the remaining text wraps around to the left side of the screen in the 40-column version of PIE Writer.}

Now, print a copy of your letter to see how well you have done.

Other FORMAT commands can be used to alter the appearance of your printed letter:

.ls—*Line spacing.* Your letter will come out single-spaced unless you change the default of 1 with a .ls command. For instance, double-spaced output can be achieved with a .ls 2 command.

.bl—*Force 'n' blank lines.* If you wish to reserve a certain amount of space in your letter, up to one full page of text, you can reserve that space for text, pictures, or graphics that will be inserted later with a .bl command. The .bl works just like the .sp command, except that the .bl forces 'n' blank lines even if the letter must break to the new page first.

.pl—*Page length.* This command gives you the opportunity to change the page length from the default of 66 to a length of 'n.'

.bp—*Begin page.* The .bp command tells FORMAT to begin the text that follows the command on a new page.

.ne—*Lines needed to bottom.* The .ne command makes sure that if there is not enough room for at least 'n' lines of text at the bottom of the page, then the text that follows the .ne begins on the next page.

FORMAT Command Review	Function
.ls n	Set line spacing
.bl n	Force 'n' blank lines
.pl n	Set page length
.bp n	Begin page
.ne n	Lines needed to bottom

If you were to alter the text in your text file, perhaps by formatting the file to print double-spaced instead of single-spaced, you would have to return to the Text Editor, insert the desired dot commands, save the file, summon the Text Processor, and review the formatted file again to ensure that the vertical spacing was correct.

A formatted copy of your file, printed using the Text Processor (see Lesson 2 of this chapter), and a literal copy, printed using the CP are reproduced below. Check them to make sure you understand all the layout commands covered in this section.

```
Universal Machines, Inc.
1000 Main Street
Paris, Texas 52500

January 1, 1999
```

```
Universal Parts, Inc.
9999 Forest Avenue
Rome, New York 99999

Dear Ladies or Gentlemen;

     Enter a paragraph here.

     Another paragraph.


                                        Yours truly,


                                        Your Name
```

```
.PO5
.SP10
.NF
.IN33
UNIVERSAL MACHINES, INC.
1000 MAIN STREET
PARIS, TEXAS  52500
.SP
JANUARY 1, 1999
.IN
.SP4
UNIVERSAL PARTS, INC.
9999 FOREST AVENUE
ROME,  NEW YORK  99999
.SP
DEAR LADIES OR GENTLEMEN;
.PP
ENTER A PARAGRAPH HERE.
.PP
ANOTHER PARAGRAPH.
.NP6
.IN33
YOURS TRULY,
.SP4
YOUR NAME
```

LESSON 5: INDENTATION

Extract Indent

To set off a quotation or other extracted material from the rest of a document, you may want to indent both the left and right margins an equal amount.

To indent each margin, for instance, 5 spaces, you must enter **.in 5** to indent the left margin and **.ll-5** ("minus" 5) to reduce the line length by 5 spaces, thereby indenting the right margin. Each of these indents would be cancelled and normal text margins would be resumed, of course, by entering a **.in** and **.ll** command, returning the indent to 0 and the line length to 65, their default values.

The following commands would have to be entered to print an extract:

```
.IN 5                          Set left and right margins
.LL -5
.SP                            Allow 1 line space
ENTER EXTRACT OR QUOTE.        Enter extracted text
.IN                            Resume normal left
.LL                               and right margins
.SP                            Allow 1 line space
ENTER NORMAL TEXT AGAIN.       Resume normal text
```

FORMAT Command Review	Function
.in n	Set indent (adjust left margin)
.ll n	Set line length (adjust right margin)

List and Outline Indent

The 'n' variable that accompanies most FORMAT commands allows you to enter any value, including a "plus" or "minus" value. For instance, when using the indent command (**.in**) immediately preceding a line you want indented, such as a list or outline entry:

- **.IN 5** *indents the text that follows it 5 spaces to the right of the page offset (***.po***)*
- **.IN+5** *indents 5 spaces to the right of the current indent*
- **.IN-5** *moves the indent 5 spaces to the left of the current indent*

.IN +5 → *Using "plus" indentations,*
.IN +5 → *you can indent*
.IN +5 → *outlines*
.IN+5 → *all the way*
.IN+5 → *across the page.*
.IN-5 ← *Using "minus" indentations,*
.IN-5 ← *you can shift*
.IN-5 ← *.indentations*
.IN ← *back to the left.*

An indent without a plus or minus sign is set to the column number given. A "plus" indent is always added to the current indent, set originally by an indent (**.in**), or temporary indent (**.ti**) command. A "minus" indent is subtracted from the current indent.

For instance, if you wanted to indent an outline entry 5 spaces, in addition to the 10 space indent already set, then run a list of items, you could use the **.ti** command as shown below:

```
.IN 10                          Set 10 space indent
.TI +5                          Indent first line an additional
THE .TI COMMAND ADDS A             5 spaces
TEMPORARY INDENT, INDENT-
ING THE FIRST LINE OF
THIS PARAGRAPH 5
ADDITIONAL SPACES.  THE
REMAINDER OF THE LINE IS
FLUSH LEFT ON A 10-SPACE
INDENT.

.TI -3                          Set indent to "hang" number of
1. THIS COMMAND ALLOWS             list item 1.
YOU TO "HANG" THE NUMBER
OF A LIST ITEM AND ALIGN
THE ITEM ITSELF AT LEFT
WITH TEXT ABOVE AND
BELOW.
```

The **.in**, **.ti**, **.pi**, and **.pp** commands can be accompanied by plus and minus values.

As discussed previously, the **.pp** command has a built-in temporary indent equal to a **.ti +5** command:

```
.PP                                  Indent first line an additional
WHEN YOU INDENT A                      5 spaces
PARAGRAPH WITH THE .PP
COMMAND, THE RESULT WILL
BE THE SAME AS IF YOU HAD
USED THE .TI+5 COMMAND.
```

You can enter a **.pi** command to change the first-line indent every time the **.pp** is used:

```
.IN 10                               Set 10-space indent
.PI 10                               Set temporary indent
.PP                                  Begin paragraph
BECAUSE .PI AND .IN HAVE
EQUAL VALUES, THIS
PARAGRAPH WILL NOT SHOW
AN ADDED FIRST-LINE
INDENT.  THE ENTIRE
PARAGRAPH WILL BE ON A
10-SPACE INDENT.
```

To indent the first line of the paragraph 10 additional spaces, you would have to change the **.pi** to **.pi +10**.

You can produce items whose numbers "hang" with **.in**, **.pi**, and **.pp** commands also:

```
.IN 13                               Set 13-space indent
.PI -3                               Hang number or "outdent" first
.PP                                    line 3 spaces
1. THESE COMMANDS WILL
MAKE THE ITEM NUMBER HANG
AND SET THE ITEM,
BEGINNING WITH THE WORD
"THESE," FLUSH LEFT ON A
13-SPACE INDENT.
```

The actual indentation from the left margin to the number 1 is 10 spaces (or, 13 minus 3).

To indent the same distance from the right margin, reduce the line length:

```
.LL -10                          Reduce the line length
.IN 13                           Set 13 spaces indent
.PI -3                           "Outdent" first line 3 spaces
.PP
1. THIS PARAGRAPH WOULD
HAVE A HANGING ITEM
NUMBER, AN ITEM FLUSH
LEFT ON A 13-SPACE
INDENT, AND RAGGED RIGHT
ON A 10-SPACE INDENT FROM
THE RIGHT.
```

FORMAT Command Review	Function
.ti n	Set temporary indent for next line

The final FORMAT unit covers format commands for producing running titles, page numbers, and centered, underlined, capitalized, or boldface text.

LESSON 6: RUNNING TITLES, PAGE NUMBERS, MARGINS, AND FINISHING TOUCHES

Running Titles

You can create running titles on every page of your document. Running heads often contain titles that reflect a page's content, along with a page number. Other pertinent information, such as a date, can be included as well, as shown below.

1999 RUNNING HEAD 100

You can create a running head with the **.he** command:

.he—*Head title.* This command prints the title, which is entered on the same line after the command, one line from the top of every page of your document. To change the title used in the running head, enter a new .he command, then the new title, and that running head will appear on the second line of the next page.

To produce a running title at the foot of your page, use the **.fo** command:

.fo—*Foot title.* This command prints whatever text that is entered on the same line after the command, three lines from the bottom of every page of your document.

The **.tl** command allows you to insert a title anywhere on the text page:

.tl—*Flush left, center, flush right.* The title that follows the .tl command is printed once, exactly where it is entered in the text.

All three of these commands, the **.he**, **.fo**, and **.tl**, follow a certain pattern for title entry. Up to three "parts" of the running head can be specified, with each part separated by an apostrophe. Their position with the running head is shown below:

Part Part 2 Part 3

Part 1 begins at the left margin. Part 2 is centered. Part 3 is aligned flush right.

To enter a three-part head title, you would enter, all on the same line:

1. *The* **.he** *dot command*
2. *The first apostrophe*
3. *Part 1 of the running head*
4. *The second apostrophe*
5. *Part 2 of the running head*
6. *The third apostrophe*
7. *Part 3 of the running head*
8. *The fourth apostrophe (optional)*

On the screen, it would look this way:

```
.HE 'PART 1'PART 2'PART 3'
```

To produce a three-part head title

To enter the same title at the foot of the page, enter:

```
.FO 'PART 1'PART 2'PART 3'
```

To produce a three-part foot title

To use only the first part of a title, just enter the first part:

```
.HE 'PART 1'
```

To produce the first part of a head title

To use only Parts 2 and 3, enter:

```
.HE 'PART 2'PART 3'
```

To produce only parts 2 and 3 of a head title

The second consecutive apostrophe indicates that Part 1 is to be left blank. Part 2 will be output in its usual place at the center of the page, and Part 3 will be flush right.

To use only Part 3 (for example, to make sure that a letter's date is right-justified), enter:

```
.TL '''JANUARY 1, 1999      To produce only the
                                third part of a head title
```

FORMAT Command Review	Function
.he t	Set head title 't'
.fo t	Set foot title 't'
.tl t	Set 't' flush left, center, flush right

Page Numbers

You can have FORMAT insert page numbers for you, on every page of your document, by using the "%" sign. To have the current page number produced at the foot of every page in your file, follow the format below:

```
.FO ''%          To center your document
                   page number at the foot of
                   the page
```

FORMAT automatically replaces % with the current page number. The command is continuous, so this one entry is all you need to assign page numbers to the whole file automatically, from page 1 to the end of your document. Of course, you can have your page number appear flush left, centered, or flush right at the head or foot of every page.

Margins

With margin commands, you can move head and foot titles up or down on the page. There are four top and bottom margins that you can manipulate to regulate this movemnent.

.ml—*First top margin (to head title).* This command specifies the number of lines from the top of the page down to, and including, the head title. The default value for .ml is n = 2.

.m2—*Second top margin.* This command specifies the number of lines between the head title and the first line of text, excluding the first line of text. The .m2 default is n = 2.

.m3—*First bottom margin (to foot title).* The third margin specifies the number of lines between the last line of text and the foot title, excluding the foot title. The .m3 default is n = 1.

.m4—*Second bottom margin.* The fourth margin sets the number of lines from the bottom of the page, up to, and including, the foot title. The second bottom margin default is n = 3.

These four margins represent the space indicated below on a printed page:

M1

```
        Part 1              Part 2              Part 3
```

M2

```
        text  text  text  text  text  text  text
        text  text  text  text  text  text  text
        text  text  text  text  text  text  text

        text  text  text  text  text  text  text
        text  text  text  text  text  text  text
        text  text  text  text  text  text  text
```

M3

```
                          -18-
```

M4

To produce a three-part running head at the top of the page and a centered page number at the foot of the page, you would enter:

```
.HE 'PART 1'PART 2'PART 3      Set the running head
.FO ''-%-                      Set automatic page
.M1 4                          numbering
.M2 4                          Set margins
.M3
.M4 4
ENTER A FULL TEXT PAGE,        Enter text
WITHOUT RUNNING HEAD AND
PAGE NUMBER HERE.
```

The entry above produces the first page of a report with margins as illustrated below:

```
                                                    4 total lines here
Part 1                    Part 2

                                                    4 blank lines here

text  text  text  text  text
text  text  text  text  text
text  text  text  text  text

                                                    1 blank line here, the
                -60-                                  default value
                                                    3 lines below the page
                                                      number to the end of
                                                      the sheet
```

Of course, you can also set margins without using head or foot titles. Just turn off Margin 1 and Margin 4 by setting them at zero, which inhibits the output of head or foot titles. Set Margin 2 and Margin 3 wherever you want them.

```
.M1 0                                   Ignore this margin
.M2 6                                   Set margins 2 and 3 at 6
.M3 6                                     each
.M4 0                                   Ignore this margin
TEXT TEXT TEXT TEXT                     Enter text
TEXT TEXT TEXT TEXT
TEXT TEXT TEXT TEXT
```

The commands entered above will produce 6 lines of space above and below the document text.

If you have no head and foot titles, you can produce a report with 4 blank lines at the top and bottom of each page by entering no margin specifications at all.

FORMAT Command Review	Function
.m1 n	Set first top margin (to & including head)
.m2 n	Set second top margin
.m3 n	Set first bottom margin (to foot)

.m4 n	Set second bottom margin (including foot)

Finishing Touches

You can center and capitalize text anywhere on the page with the **.ce** and **.cp** commands:

.ce—*Center.* This command tells FORMAT to center the following 'n' lines of unformatted text. Whether it is used in the fill (.fi) or the top no-fill mode (.nf), the .ce command centers text broken line by line.

.cp—*Capitalize.* To capitalize for emphasis within text, use the .cp command, followed by the 'n' lines of text you want capitalized. This command is useful for entering subheads or for capitalizing a single word in the middle of a sentence, as shown below:

```
.FI                          Set the fill mode
THIS IS THE WAY TO           Enter text
.CP                          Capitalize the word on
CAPITALIZE                     the next line
TEXT                         Return to normal text
```

If your printer supports it, you also can use the boldface and underline commands, **.bf** and **.ul**:

.bf—*Boldface.* The printer backspaces and restrikes text on the 'n' lines or changes to boldface font following the .bf command to achieve boldface text. As with the .cp command, the .bf can be used to affect a single word in a sentence, making it bold.

.ul—*Underline.* To underline text, precede the 'n' lines of text you want underlined with the The .ul command is capable of underlining even a single word in a sentence.

You may need to reconfigure FORMAT to use these commands with your printer (see Chapter 4 for details).

Note that the **.cp**, **.bf**, and **.ul** commands affect the entire line (or lines) that follow the command. To have these commands work on only a few words with a line, those words must be set off as a separate line when the text is entered. In fill mode, however, when those lines are formatted, they will take their proper place within the running text.

The following example demonstrates how all four of these commands can work together:

```
.NF                                    Set no-fill mode
.CE 2                                  Center the next 2
CENTER THIS LINE.                        unformatted lines
.CP 3                                  Capitalize the next 3
CENTER AND CAP THIS LINE.                unformatted lines
.BF                                    Boldface the next line
BOLDFACE AND CAP THIS LINE.
.UL                                    Underline the next
UNDERLINE AND CAP THIS LINE.             line
```

FORMAT Command Review	Function
.ce n	Center the next 'n' lines
.bf n	Print the next 'n' lines in boldface
.ul n	Underline the next 'n' lines
.cp n	Capitalize the next 'n' lines

Read and work through the explanations of the commands covered until you have them mastered, then use the Reference Card to refresh your memory from time to time. All of the FORMAT commands, including those covered in this section and the more sophisticated commands not yet covered, are documented in Chapter 5, "Reference Section."

Chapter 4, "PIE and FORMAT Configure," discusses how to set up the PIE Writer system permanently to accommodate your normal printer and page appearance needs. You do not need to change any of the default values in PIE Configure or FORMAT Configure, however, to begin creating and printing error-free documents.

4

PIE and FORMAT Configure

The system provided with each PIE Writer package, the diskette and documentation, is a complete word processing system that operates smoothly with almost any printer or peripherals you may own.

The only information missing from the PIE Writer system is your computer's hardware configuration and printing needs. More precisely, PIE Writer needs to know specifics about what printer you are using, what page appearances meet your typical needs, and so on. The configuration files supplied on your PIE Writer diskette, "PIE CONFIGURE" for the editor (choice **3** on the System Menu) and "FORMAT CONFIGURE" for the text formatter (choice **4**), allow you to customize the system to match the hardware you are using and the appearances you prefer, both in editing and printing.

Do not reconfigure the original PIE Writer system diskette. Use the backup of the system diskette that you made after booting the original diskette the first time to reconfigure PIE and FORMAT.

Throughout this section, look at the options displayed on the screen as you read along in the documentation.

PIE CONFIGURE

When you wish to inspect or alter your present PIE configuration, select option **3** from the main System Menu. This copies the PIE program code to computer memory and allows you to review PIE Configure's default values, the values that PIE Writer will use unless you change them.

After you have chosen option **3** from the System Menu, the following screen appears:

```
1 - PRINTER INTERFACE
2 - LOWER CASE AND SPECIAL CHARACTERS
3 - OTHER PIE EDITOR DEFAULTS
4 - PROGRAM DISK SLOT, DRIVE, VOLUME
5 - DOUBLETIME SPOOLER OPTION

DO YOU WANT TO LOOK AT OR MODIFY VALUES
IN ONE OF THE ABOVE CATEGORIES?
```

{Choice 5 is only available for the Apple II and Apple II Plus.} If you respond **N** (for no) to the prompt, you are transferred to the main System Menu. If you respond **Y** (for yes), you receive another prompt:

```
WHICH ONE?
```

Instead of responding with **Y** or **N**, you can simply press the number corresponding to the line whose series of default values you wish to alter. Then, when prompted, enter the new value or select a new setting from the choices presented.

Printer Interface

Select **1** to view the printer interface options:

```
1 - PRINTER SLOT.............1
2 - PRINTER INITIALIZATION....NONE
3 - HIGH BIT SET ON OUTPUT....YES
4 - LINE FEED AFTER RETURN....NO

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?
```

Before responding to the prompt above, read the descriptions of each option below:

```
1 - PRINTER SLOT.............1
```

As shipped, the PIE Command Processor assumes that your printer interface card has been installed correctly in slot 1. If for some reason your interface card is not installed in that slot, you can reconfigure PIE to alert the system that the card is in another slot. Enter any number from **2** through **7** to change the default.

```
2 - PRINTER INITIALIZATION....NONE
```

If your printer interface card or printer requires a character or character sequence initialization, you may enter up to seven characters here. To enter a control character into the initialization string, prefix it with **CTRL-SHIFT-M** (Apple //e and Franklin: use **CTRL-]**). Each time text is output from the Command Processor to the "printer slot" (defined above), these characters are sent to the printer just before the actual text.

```
3 - HIGH BIT SET ON OUTPUT....YES
```

Some printer also require that the high-order bit in every 8-bit byte (bits and bytes are elements in the computer's machine language) be set on output for proper operation. PIE'S default setting covers this circumstance. Type **N** (for no) to change the default. This option is in effect for output to any slot or address from the the Command Processor (not just the "printer slot").

```
4 - LINE FEED AFTER RETURN....NO
```

Finally, some printers cannot produce line feeds after carriage returns unless they are instructed to do so by the software. PIE assumes that your printer or printer interface card can handle line feeds after carriage returns on its own. If the Command Processor prints out text all on one line, type **Y** (for yes) to instruct PIE to send line feeds to the printer.

The line feed option is in effect for output to any slot or address from the the Command Processor (not just the "printer slot"). This option may be temporarily changed using the **%LF** command in the Command Processor.

When you are finished making printer interface value changes, type **N** (for no) in response to the prompt to return to the PIE Configure menu.

Lowercase and Special Characters

Select **2** to view the lowercase and special character options:

```
1 - OUTPUT LOWER CASE TO CRT..NO
2 - INITIAL LOWER CASE INPUT..YES
3 - SHIFT KEY MOD.............NONE
4 - HIGH BIT SET IN BUFFER....YES
5 - CONVERT BREAK LINE TOKEN..Z

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?

1 - OUTPUT LOWER CASE TO CRT..NO
```

The original Apple II and Apple II Plus video display cannot show lowercase letters properly. Standard uppercase letters are displayed in white on a black background to represent lowercase. "Inverse" characters, black letters on a white field, represent uppercase letters. If your system includes a lowercase adapter, you must change this PIE default. Type **Y** (for yes) to display true uppercase and lowercase characters. {If you are using an 80-column board, or have an Apple //e or Franklin, this option has already been set to YES by the System Configure program.}

```
2 - INITIAL LOWER CASE INPUT..YES
```

Keystrokes are normally entered as lowercase letters, with uppercase accessed by using the → key {Apple //e and Franklin use the **SHIFT** key}. Type **N** (for no) to enter letters initially in uppercase and access lowercase letters with the same key.

```
3 - SHIFT KEY MOD.............NONE
```

Select **3** to receive the display below:

```
0 (NONE), 1 (GAME I/O), 2 (ROMPLUS) OR
3 (U/L KBD)?
```

There is hardware modification that can be made to the original Apple II and Apple II Plus that allows the **SHIFT** keys to function "normally," as on a regular typewriter. Even if the hardware modification is made, it will not be recognized by PIE Writer unless the proper setting is made here. There is a detailed discussion of this hardware modification in Chapter 6. Type **1** for GAME I/O, **2** for ROMPLUS (and the slot of the ROMPLUS card), and **0** for no modification. Type **3** if you have a full upper/lower case keyboard (such as an Apple II Plus with a "keyboard enhancer"). {If you have an Apple //e or Franklin, option 3 has already been selected by the System Configure program.}

```
4 - HIGH BIT SET IN BUFFER....YES
```

This indicates whether the high-order bit of each character in the text buffer is set to 1 or 0. For all but special programming applications, use the default (yes). Change the default by typing **N** (for no).

```
5 - CONVERT BREAK LINE TOKEN..Z
```

When loading a file that is known to have or may have lines longer than the text editor line length of 64 {128 for 80-column versions}, you can use a Command Processor auxiliary command (**%CB**) to shorten the lines in a file to a specific column

value. When the lines are shortened, a CTRL-Z is inserted at the end of each broken line. Enter any control character here to change the break line token.

When you have finished making changes to the lowercase and special character options, type **N** (for no) in response to the prompt to return to the System Menu.

Other PIE Editor Defaults

Select **3** to view the other PIE Editor default options:

```
1 - PIE EDIT BUFFER END.......$95FF
2 - INITIAL BELL COLUMN.......38
3 - INITIAL TAB INCREMENT.....8
4 - PPWRAP INITIALIZATION.....OFF

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?

1 - PIE EDIT BUFFER END.......$95FF
```

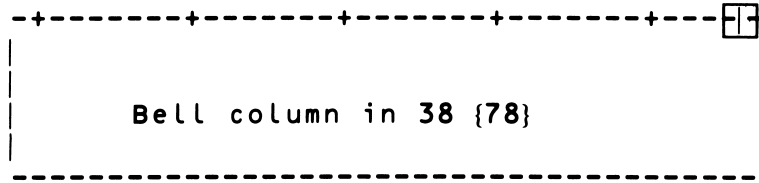
In order for you to enter text characters into the computer, an area of the computer's storage memory, the text buffer, must be reserved for character storage. On very long documents, you can "fill up" the buffer area by continuing to key in characters until all of the reserved space has been filled. When the text buffer is filled, the Status Line displays an "ALMOST OUT OF MEMORY" message, indicating that it is time to save a copy of your file on diskette.

The text buffer begins at $4000 {$4C00 for 80-column versions} and normally ends at the highest available memory location below the Disk Operating System (DOS), which is $95FF. You may want to decrease this value by entering another hexadecimal address if you need to reserve space for a large printer driver or other utility program.

The majority of users will never need to alter this setting.

```
2 - INITIAL BELL COLUMN.......38
```

If you were using the Text Editor right now, you would see the bell column marker, a reverse dash, in column 38 {column 78 for 80-column versions}, where it is indicated below:



The bell column marker indicates the position in the line at which either of the following occurs:

1. *In Manual Mode, the bell tone beeps to indicate that the cursor has reached the end of the first window. The bell tone beeps again at the end of the right-hand side of the window simply to remind the operator to press* **RETURN**.
2. *In PPWrap Mode, the bell column will not produce the bell tone. Instead, it will move the cursor down to the next line, positioned beneath the first printable character of the previous line. If this occurs in the middle of the word, all characters back to the last space character will be moved to the next line as well.*

Enter any value from **1** to **64** {or **128** for 80-column users}, then press **RETURN** to set a new bell column marker.

```
3 - INITIAL TAB INCREMENT.....8
```

Initially, the tab markers are set 8 spaces apart, beginning with the first column. Enter any value from **1** to **64** {or **128** in 80-column versions} to change this setting from the default to one you prefer, then press **RETURN**.

```
4 - PPWRAP INITIALIZATION.....OFF
```

Select **4** to receive the display below:

```
0 (OFF), 1 (PPWRAP), 2 (INDENT)?
```

As shipped, PIE Writer's Wrap mode is turned off. To change the default (0), type **1** for on, or **2** for indent mode (see Chapter 5 for a full explanation of the auto indent feature).

When you have finished changing as many PIE Editor defaults as you wish, type **N** (for no) to return to the PIE Configure main menu.

Select **4** to view the diskette slot, drive, and volume options:

```
1 - SLOT.....6
2 - DRIVE.....1
3 - VOLUME.....0

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?
```

The slot, drive, and volume options are for users with more than one drive. These options tell the system where to find the system diskette. {For 48K versions of PIE, Writer, the system diskette must always be in the slot, drive, and volume specified here whenever you transfer between PIE and FORMAT or when the auxiliary files are accessed.}

```
1 - SLOT.....................6
```

If the disk interface card attached to the drive you are using for your system diskette does not occupy slot 6, type the number of the slot in which the card is installed, from **1** to **7**.

```
2 - DRIVE....................1
```

If you wish to use DRIVE 2 for your system diskette, change this option by typing **2**.

```
3 - VOLUME...................0
```

The volume should be set to 0 unless you have a hard disk system that requires a volume specification.

Type **N** (for no) in response to the prompt when you have finished changing system defaults for diskette slot, drive, and volume.

Doubletime Spooler Option

Select **5** if you have purchased the Doubletime Spooler:

```
DOUBLETIME SPOOLER...........NO

DO YOU WANT TO CHANGE THE ABOVE VALUE?
```

{This option is only available for the Apple II and II Plus.} Type **Y** (for yes) in response to the prompt above if you are using the Doubletime Spooler. You must have transferred the "Doubletime Printer" file from that package onto your system diskette to use the spooler.

Saving PIE Configure Options

Those are all the PIE Configure options. Change the appropriate settings to suit your typical needs, then answer **N** (for no) to the prompt:

```
DO YOU WANT TO LOOK AT OR MODIFY
ADDITIONAL VALUES IN ONE OF
THE ABOVE CATEGORIES?
```

Another prompt appears:

```
DO YOU WANT TO PERMANENTLY
UPDATE PIE?
```

Answer **Y** (for yes) to store an updated copy of PIE on the system diskette. Your changes are "permanent" in that the system will begin each session with whatever values you select. The next time you boot the system, it will remember your preferred values. You can change these permanent settings, however, by selecting the PIE Configure option on the System Menu as often as you wish.

FORMAT CONFIGURE

This section of the PIE Writer program allows you to set up FORMAT to take advantage of your printer's specific features.

Make sure that you are about to reconfigure your backup—not your original—system diskette, then select **4** in response to the System Menu prompt. The following screen appears:

```
1 - PRINTER INTERFACE
2 - LOWER CASE AND SPECIAL CHARACTERS
3 - PAGE FORMAT (SPACING, LINE LENGTH)
4 - TOP AND BOTTOM MARGINS
5 - PARAGRAPHS AND FILLING
6 - PROGRAM DISK SLOT, DRIVE, VOLUME

DO YOU WANT TO LOOK AT OR MODIFY VALUES
IN ONE OF THE ABOVE CATEGORIES?
```

Again, walk through the options on your screen while reading along in the documentation.

Printer Interface

Select **1** to view the printer interface options:

```
1 - PRINTER SLOT..............1
2 - PRINTER TYPE..............OTHER
3 - PRINTER INITIALIZATION....NONE
4 - HIGH BIT SET ON OUTPUT....YES
5 - LINE FEED AFTER RETURN....NO
6 - UNDERLINE CAPABILITY......BS
7 - BOLDFACE CAPABILITY.......BS
8 - PAGE STOP ENABLED.........NO

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?


1 - PRINTER SLOT..............1
```

If you have a standard interface card installed in slot 1, FORMAT Configure is set up correctly for your system. If you do not have a standard interface card or if your card is not in slot 1, then type **1**, and the prompt below appears:

```
DO YOU HAVE A STANDARD
APPLE II INTERFACE CARD?
```

Respond by typing **Y** (for yes) if you have a standard card. The next prompt asks you enter the card's slot number:

```
IN WHICH SLOT #
```

Type the slot number, from **1** through **7**, and FORMAT Configure will adjust the printer driver address for you.

If you want to use your own custom printer driver, instead of going through the standard firmware on the card, respond **N** (for no) to the first prompt. Then enter the printer driver address in response to the prompt below, and press **RETURN**:

```
ADDRESS OF YOUR PRINTER DRIVER?
```

The "Address Tables" in Chapter 6 provide information on where to locate user-written printer drivers.

```
2 - PRINTER TYPE.............OTHER
```

As shipped, PIE Writer is ready to function as a word processing system with most printers commonly available on the market. FORMAT can be reconfigured to take advantage of your printer's maximum capabilities, however. Type **2** to receive the display below:

```
1 - DIABLO  1620          4 - EPSON MX-80
2 - QUME SPRINT 5         5 - CENTRONICS 737
3 - NEC SPIN 5510         6 - OTHER

SELECT PRINTER TYPE:
```

If your printer is one of the ones listed, type the number to the left of the printer's name in response to the prompt. FORMAT Configure alters the remaining five values (items 3 through 7) in the printer interface option, allowing PIE Writer to take maximum advantage of your printer's capabilities. Users experienced with these printers may override any of the six values set up after the printer type is selected.

Selecting one of the first three printers displays the prompt:

```
# OF CHARACTERS PER INCH:
```

Enter the appropriate number and press **RETURN** to reconfigure FORMAT for any of those three printers.

If **4** is selected (Epson MX-80), FORMAT Configure displays the prompt:

```
0 (ORIG MX-80), 1 (WITH GRAFTRAX-80),
2 (WITH GRAFTRAX-PLUS)
```

Select the appropriate option. Note: all MX-80 printers manufactured after March 15, 1982 have Graftrax-Plus ROM's standard.

When the sixth printer option is chosen, FORMAT Configure provides the prompt below:

```
CAN YOUR PRINTER BACKSPACE?
```

Respond by typing **Y** (for yes) or **N** (for no). FORMAT sets the sixth and seventh printer interface parameters based on your answer.

Again, the remaining five printer interface defaults can be overridden to accommodate your printing needs.

```
3 - PRINTER INITIALIZATION....NONE
```

If your printer interface card or printer requires a character or character sequence initialization, you may enter up to seven characters here. To enter a control character, prefix it with **CTRL-SHIFT-M** {Apple //e and Franklin: use **CTRL-]**}. Each time FORMAT begins output to a printer, these characters are sent to the printer just before the actual text.

```
4 - HIGH BIT SET ON OUTPUT....YES
```

Some printers require that the high bit be set for proper operation. FORMAT's default setting assumes this. Type **N** (for no) to change the default.

```
5 - LINE FEED AFTER RETURN....NO
```

This setting varies from printer to printer. If FORMAT prints out the document all on the same line, then the answer should be **Y** (for yes). Many printer interface cards automatically provide a line feed when the carriage return character is passed.

```
6 - UNDERLINE CAPABILITY.......BS
```

Type **6** to receive the display below:

```
0 (NONE), 1 (BS), 2 (S/S), 3 (OVRLAY)?
```

Type **0** if your printer cannot underline text using any of the methods described below.

BS indicates that FORMAT will implement the **.ul** command with a combined "backspace" and then "_" to create an underline. Type **1** to select backspace.

S/S indicates that your printer has an automatic underline capability (such as on the Centronics 737). Start/stop underlining is initiated with the control character sequence you define after typing **2** to receive the next prompt:

```
(ENTER CTRL CHARS USING CTRL-SHIFT-M)

UNDERLINE START SEQUENCE:
UNDERLINE STOP SEQUENCE:
```

To enter a control character, prefix it with **CTRL-SHIFT-M** {Apple //e and Franklin: use **CTRL-]**}. Press **RETURN** after typing the underline sequence.

OVRLAY is used with printers such as the original Epson MX-80, which cannot backspace or use start/stop, and is selected when you type **3** for this option.

AN ESCAPE Character is entered by CTRL-[[

```
7 - BOLDFACE CAPABILITY.......BS
```

Type **7** to receive the display below:

```
0 (NONE), 1 (BS), 2 (S/S)?
```

Type **0** if your printer cannot produce boldface copy using the two methods described. Type **1** if you selected BS in the underline capability option. Type **2** if your printer uses a start/stop sequence to produce boldface print. You are prompted for the start and stop sequences. To enter a control character, prefix it with **CTRL-SHIFT-M** {Apple //e and Franklin: use **CTRL-]**}.

```
8 - PAGE STOP ENABLED.........NO
```

Some printers require or allow single-sheet paper. If you are using single-sheet paper, you want the output to halt after each page is printed so that you can feed a new sheet in. Select **Y** (for yes) to accomplish this.

When you are finished reconfiguring the printer interface, return to the main FORMAT Configure menu by answering **N** (for no) in response to the DO YOU WANT... prompt.

Lowercase and Special Characters

Select **2** on the main FORMAT Configure menu to reconfigure the way lowercase and special characters are entered on the keyboard and appear on the screen.

```
1 - OUTPUT LOWER CASE TO CRT..NO
2 - INITIAL LOWER CASE (.GL)..YES
3 - SHIFT KEY MOD.............NONE
4 - ESCAPE CHARACTER .........']'
5 - BLOCK TERMINATING CHAR....'<'
6 - LF AFTER CR OUTPUT TO DISK.NO
7 - STOP CHAR OUTPUT TO DISK..C

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?
```

You can reconfigue each of the following options by selecting the appropriate number.

```
1 - OUTPUT LOWER CASE TO CRT..NO
```

If you have a lowercase adapter, enter **Y** for yes in response to the prompt. This option should be set to the same value chosen in PIE Configure. {If you have an 80-column version, or are using an Apple //e or Franklin, this has already been set up for you by the System Configure program.}

```
2 - INITIAL LOWER CASE (.GL)..YES

3 - SHIFT KEY MOD.............NONE
```

These two options refer to text entry via the **.gl** command. They should be set to the same values as those chosen in PIE Configure.

```
4 - ESCAPE CHARACTER..........']'
```

Typing the FORMAT escape character while in PIE allows you to enter special format commands within a file. You can change the escape character from the default (]) to any character you prefer.

```
5 - BLOCK TERMINATING CHAR....'<'
```

The block terminating character is entered at the end of the a data block in a secondary file when merging files to produce form letters, mailing labels, and so on. The '<' character is the default. Enter any character you prefer to change the default.

```
6 - LF AFTER CR OUTPUT TO DISK.NO
```

As stated previously, some printers rely on software to produce line feeds with each carriage return after a printed line. If this is the case with your printer, and you intend to print a file after first writing it to diskette, change the default by **Y** (for yes). Files printed from a diskette will have the line feed embedded automatically after each carriage return.

```
7 - STOP CHAR OUTPUT TO DISK..C
```

The printer stop escape,]+, must be cancelled if outputting a file to diskette. The]+ is translated to CTRL-C, which is recognized by the Doubletime Spooler software.

To return to the main FORMAT Configure menu after reconfiguring the lowercase and special character settings, type **N** (for no).

Page Format

To change the default format of your page, including spacing and line length, select **3** from the main FORMAT Configure menu to reveal the screen below:

```
1 - LINE SPACING.............1
2 - PAGE LENGTH..............66
3 - LINE LENGTH..............65
4 - PAGE OFFSET..............0
5 - INITIAL OFFSET CHAR.......' '

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?

1 - LINE SPACING.............1
```

The default is set for single-spaced printouts. Select **2** for double-spaced or **3** for triple-spaced reports. You can enter any number up to one less than the page length, then press **RETURN**.

```
2 - PAGE LENGTH..............66
```

FORMAT assumes an 8 1/2 x 11-inch report page, and 6 printed lines per vertical inch. Enter any value you want here from **1** to **250**, then press **RETURN**.

Note that this value is equal to the number of lines that can be printed on one of your pages, not necessarily the number of lines you want printed.

```
3 - LINE LENGTH..............65
```

FORMAT assumes a 65-character line based on a 6 1/2-inch line and a 10 character-per-inch printer. The same line length on a 12 cpi printer requires a 78-character line. Enter whatever value suits your needs, from **1** to **132**, then press **RETURN**.

```
4 - PAGE OFFSET..............0
```

Set the page offset value appropriate for your printer permanently here (see Chapter 3 to determine that value). Any value from **1** to a maximum of one less than the line length value is valid. Enter the page offset and press **RETURN**.

```
5 - INITIAL OFFSET CHAR.......' '
```

If you want to precede every line in your file with a character that your printer recognizes, enter the character here, and press **RETURN**. The default value is a blank space, for normal use. This character is output only if the page offset value is greater than 0.

Type **N** (for no) in response to the prompt when you have altered all the page format defaults you want.

Top and Bottom Margins

FORMAT Configure lets you change the head and foot margins permanently. Select **4** to see the following screen:

```
1 - MARGIN 1 VALUE.............2
2 - MARGIN 2 VALUE.............2
3 - MARGIN 3 VALUE.............1
4 - MARGIN 4 VALUE.............3

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?
```

Type the number of the margin you want to alter to change the default setting. The areas on the printed page that correspond to each margin value are:

m1 = the number of lines from the top of the page down to, and including, the head title
m2 = the number of lines between the head title and the first line of text
m3 = the number of lines between the end of the text and foot title
m4 = the number of lines at the bottom of the page, including the foot title

Alter the margin values according to your specific needs, then press **RETURN**. Although the maximum value for each margin is one less than the current page length value, normally they are much smaller. Remember that the number of text lines available on the page is equal your page length value minus the total of all the margins.

If you do not use the head title (**.he**) or foot title (**.fo**) commands, 4 blank lines appear at the top and bottom of every printed page if you are using the default settings. Also, setting the first and fourth margin values to zero prevents the output of head or foot titles.

Return to the main FORMAT Configure menu by responding **N** (for no) to the prompt.

Paragraphs and Filling

Option 5 lets you change paragraph and filling modes permanently. Select **5** to see the options below:

```
1 - INITIAL FILL MODE.........YES
2 - INITIAL RIGHT ADJUST......NO
3 - PARAGRAPH INDENT..........+5
4 - PARAGRAPH SPACING.........1
5 - PARAGRAPH NEED VALUE......2

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?

1 - INITIAL FILL MODE.........YES
```

When the fill mode is on, the FORMAT program gathers as many words entered in the PIE window as it needs to fill a specified line, so that words entered on different lines may be printed on the same line. If you are primarily using FORMAT to print already formatted output, such as spreadsheet program output or computer programs, you should change the default by responding **N** (for no).

```
2 - INITIAL RIGHT ADJUST......NO
```

Initial right adjust inserts spaces between words or characrters in a printed line so that the text is even, or justified, along the right-hand margin. If you want right-justification by default, type **Y** (for yes).

```
3 - PARAGRAPH INDENT..........+5
```

Again, the slot, drive, and volume options are for users with more than one drive, specifying where the system diskette is to be found. {For 48K versions of PIE Writer, the system diskette must be in the slot, drive, and volume specified whenever you transfer between PIE and FORMAT or access auxiliary files.}

```
4 - PARAGRAPH SPACING.........1
```

This option gives you the opportunity to change the number of blank lines inserted between paragraphs.

```
5 - PARAGRAPH NEED VALUE......2
```

The **.pn** command includes an automatic **.ne** command that "looks ahead" to see how many text lines remain on the page. If the remaining lines are fewer that the lines

"needed" for a paragraph, then a page break is forced before the new paragraph is begun. Enter whatever "need" you want invoked whenever a **.pp** or **.np** command that you have entered is encountered in a file, then press **RETURN**.

Return to the main FORMAT Configure menu after selecting appropriate paragraph and fill mode settings.

Program Disk Slot, Drive, Volume

The final option involves the use of more than one disk drive. Select **6**.

```
1 - SLOT.....................6
2 - DRIVE....................1
3 - VOLUME...................0

DO YOU WANT TO CHANGE ONE
OF THE ABOVE VALUES?
```

Again, the slot, drive, and volume options are for users with more than one drive, specifying where the system diskette is to be found. {For 48K versions of PIE Writer, the system diskette must be in the slot, drive, and volume specified whenever you transfer between PIE and FORMAT or access auxiliary files.}

Set the SLOT to the slot of your disk controller card (usually 6), the DRIVE to the drive that contains the PIE Writer diskette, and the VOLUME, if using a hard disk system, to the necessary value.

Saving FORMAT Configure Options

Those are all the FORMAT Configure options. Change the appropriate settings to suit your typical needs, then answer **N** (for no) to the prompt:

```
DO YOU WANT TO LOOK AT OR MODIFY
ADDITIONAL VALUES IN ONE OF
THE ABOVE CATEGORIES?
```

Another prompt appears:

```
DO YOU WANT TO PERMANENTLY
UPDATE FORMAT?
```

Answer **Y** (for yes), and an updated copy of FORMAT is saved on the system diskette. The next time you boot the system, it will remember your preferred values. You can change FORMAT Configure defaults as often as you wish.

You can exit the System Menu and transfer to Applesoft BASIC by typing **5** in response to the System Menu.

Chapter 5, "Reference Section," reviews basic PIE and FORMAT text commands and introduces new, advanced commands. Some of the new commands perform sophisticated word processing functions. Explore the commands in the next section when you have mastered the commands and concepts already covered.

5

Reference Section

This section contains explanations of every PIE Text Editor, PIE Command Processor, and FORMAT Text Processor command in the PIE Writer system. The basic commands have already been covered in earlier sections; however, new, advanced commands are presented here for the first time. The section ends with a discussion of error messages you may encounter. An abbreviated list of all PIE Writer commands is provided on the Reference Card.

PIE TEXT EDITOR FOR THE APPLE //E

PIE Text Editor commands are documented here in groups according to the similarity of their functions.

Within each group you will find both simple commands and compound commands described. Compound commands are formed by preceding the single-key simple commands with prefixes. All compound commands are initiated by first typing the **ESC** key; some compound commands also take a modifier, which is typed after the **ESC** key is pressed.

After **ESC** is pressed, the word "ARG:" appears on the lower right of the Status Line. ("ARG" is short for "argument".) Sometimes simply prefixing a simple command with the **ESC** key is enough to change the function of the command; just as often, arguments or modifiers are entered. The text of the modifier appears on the status line as you enter it in.

Possible modifiers include: a specific letter, some arbitrary character (indicated by *c* in the descriptions below), a number (indicated as *nn* or *mm*), or a string of characters (represented by *s*, *s1* or *s2*). The modifiers and the simple commands which they prefix are shown in regular type; the simple commands themselves are in bold face.

Simple Cursor Movement

The cursor indicates where your next typed character will appear. The cursor's column and line position are reflected in the Status Line at the bottom of the screen. With the following commands you can move the cursor to any position on the screen.

PIE Command Key	Function
↑ or **CTRL-K**	Moves cursor down one line
↓ or **CTRL-J**	Moves cursor up one line
→ or **CTRL-U**	Moves cursor one space to the right
← or **CTRL-H**	Moves cursor one space to the left

Each of the vertical cursor movement commands triggers single-line scrolling if the cursor movement would otherwise cause the cursor the leave the text window.

Cursor Jumps

PIE Command Key		Function
	CTRL-G	Toggles between moving cursor to the top left or bottom left corner of screen
	CTRL-B	Toggles between moving cursor to beginning or end of line
ESC nn	→ or CTRL-U	Moves cursor to column 'nn'
	RETURN	Moves cursor to column one of next line

CTRL-G alternates between one of two functions: it toggles back and forth between moving the cursor to the top left-hand corner and moving it to the bottom left-hand corner of the screen. From any cursor position, other than the top left-hand corner, **CTRL-G** moves first to the top left-hand corner.

CTRL-B initially moves the cursor to the end of whatever line the cursor is on, then toggles back and forth between moving the cursor to the beginning and to the end of that line.

ESC nn → or **ESC nn CTRL-U** moves the cursor to column 'nn'. Values from 1 to 64 are valid {1 to 128 for 80-column versions}.

The effect of the **RETURN** key depends on the current text entry mode (Manual PPWrap, or Indent).

In Manual mode, the **RETURN** key always performs only a cursor movement: it causes the cursor to go to the beginning of the next line. In PPwrap or Indent modes, typing **RETURN** at the end of a line inserts a blank line after the current one, and then moves the cursor to the position just below the leftmost character of the current line.

Tabbing

PIE Command Key		Function
	TAB or **CTRL-I**	Moves cursor one tab stop to the right
ESC S	TAB or CTRL-I	Sets bell column to current column
ESC C	TAB or CTRL-I	Clears bell column
	CTRL-A	Moves cursor one tab stop to the left
ESC S	CTRL-A	Sets tab marker in current column
ESC C	CTRL-A	Clears tab marker in current column
ESC 0	CTRL-A	Clears all tab markers
ESC nn	CTRL-A	Sets tab markers every 'nn' columns
ESC W	RETURN	Toggles between word and column tabbing

Attempting to tab beyond the edges of the screen has no effect, except in Word Tab mode. In Word Tab mode, instead of tabbing to the next tab stop, you can tab to the first character of the next word to the right or left: **TAB** or **CTRL-I** tabs to the word on the right, and **CTRL-A** tabs to the word on the left. If there are no more words on the line, a tab command simply moves to the first letter of the last word on the line above (**CTRL-A**) or to the first letter of the first word on the line below (**TAB** or **CTRL-I**).

ESC S TAB or **ESC S CTRL-I** sets the bell column at whatever column the cursor is in. **ESC C TAB** or **ESC C CTRL-I** clears the bell column, regardless of the cursor position and the column in which the bell is set. The bell column is indicated by a reverse video character {or vertical bar} at the top of the screen; it functions like the bell on a typewriter, causing the computer to "beep" when the cursor moves past it.

ESC S CTRL-A sets a tab at the column the cursor is in; **ESC C CTRL-A** clears a tab at the cursor. All commands which change the tab stops are effective whether or not you are in column tabbing mode. **ESC 0 CTRL-A** clears all tabs, regardless of the cursor position and the number of columns in which tabs are set. **ESC nn CTRL-A** sets a tab every 'nn' columns beginning at the first column; valid values for 'nn' are from 1 to 64 {1 to 128 for 80-column versions}.

The type of tabbing (word or column) is set separately for each of the three text entry modes (Manual, PPwrap, and Indent). Initial default settings are Word Tabbing for PPWrap mode and column tabbing for Manual and Indent modes. **ESC W RETURN** toggles the tabbing method for the current text mode. If you shift between modes, the type of tabbing previously selected in that mode is restored.

Line and Window Scrolling

	PIE Command Key	Function
	CTRL-V	Advances screen 21 lines (one page)
ESC nn	CTRL-V	Advances 'nn' pages
ESC	CTRL-V	Moves current line to top of screen
	CTRL-R	Reverses screen 21 lines (one page)
ESC nn	CTRL-R	Reverses 'nn' pages

	CTRL-N	Advances cursor and screen one line
ESC	CTRL-N	Advances cursor and screen one-half page
ESC nn	CTRL-N	Advances cursor and screen 'nn' lines
	CTRL-Y	Reverses cursor and screen one line
ESC	CTRL-Y	Reverses cursor and screen one-half page
ESC nn	CTRL-Y	Reverses cursor and screen 'nn' lines

CTRL-V advances the screen 21 lines, or one "page", and **CTRL-R** reverses the screen 21 lines. The cursor remains in its original position on the screen.

ESC nn CTRL-V advances the screen 'nn' pages; **ESC nn CTRL-R** reverses the screen 'nn' pages.

ESC CTRL-V moves the line the cursor is in to the top of the screen, homing the line. The cursor remains in the same column.

CTRL-N advances the screen one line. The cursor advances with the line it was on. **CTRL-Y** reverses the screen and cursor one line.

ESC nn CTRL-N advances the screen and cursor 'nn' lines, and **ESC nn CTRL-Y** reverses the screen and cursor 'nn' lines. **ESC CTRL-N** advances the screen and cursor one half page, and **ESC CTRL-Y** reverses the screen and cursor one half page.

Go To a Line

PIE Command Key		Function
	CTRL-T	Go to beginning of file
ESC	CTRL-T	Go to end of file
ESC nn	CTRL-T	Go to line 'nn'

CTRL-T moves the cursor and screen to the beginning of the file you are working on; **ESC CTRL-T** moves the cursor and screen to the end of that file. **ESC nn CTRL-T** moves the cursor and screen to line 'nn'.

Some commands, such as go to a line, take a few seconds to complete. PIE Writer displays a "BUSY" message on the status line during this time. Even while the "BUSY" message is being displayed, however, you can continue entering commands (or text). This is called 'type-ahead'; the computer remembers what you have typed and enters it at the cursor's original position after the first command has been carried out.

Inserting and Deleting

	PIE Command Key	Function
	CTRL-P	Begins or ends insertion mode
ESC s	CTRL-P	Inserts string 's' at cursor
	CTRL-S	Deletes character to the left and backspaces
	CTRL-E	Inserts a blank line at cursor
ESC nn	CTRL-E	Inserts 'nn' blank lines at cursor
	DELETE or **CTRL-D**	Deletes character at cursor and moves other characters one space to the left
ESC c	DELETE or CTRL-D	Deletes all characters up to but not including 'c'
ESC	DELETE or CTRL-D	Deletes all characters to the right of the cursor
	CTRL-^	Deletes word at cursor

ESC	CTRL-^	Deletes line at cursor
ESC nn	CTRL-^	Deletes 'nn' lines at cursor
ESC #mm,nn	CTRL-^	Deletes lines 'mm' through 'nn'
ESC	CTRL-B	Deletes all blank lines below cursor

CTRL-P toggles in and out of the insert mode, in which a string of characters can be inserted at the cursor. The message "INSERT MODE" appears on the Status Line to when the insert mode is active; it disappears when **CTRL-P** toggles out of it. If there is not enough space on the line for character insertion while in the insert mode, the error message "NOT ENOUGH ROOM ON LINE" will be displayed on the Status Line.

ESC s CTRL-P allows you to insert 's'—any string of characters—at the cursor, forcing all other text on the line to make room by moving to the right. **ESC CTRL-P** inserts 's'—that is, the same string—at the cursor again. The string is limited to 26 {or 58 in 80-column versions} characters, including spaces.

The **CTRL-S** key deletes text as it backspaces. This command has no effect if the cursor is in column one. In insert mode (initiated by **CTRL-P**) **CTRL-S** drags every character to the right of the cursor to the left as it deletes and backspaces.

CTRL-E inserts a blank line, and **ESC nn CTRL-E** inserts 'nn' blank lines, at the cursor. The line the cursor was on before the command moves below the inserted line.

The **DELETE** or **CTRL-D** command deletes a character at the cursor and shifts all the text to the right of the cursor to the left. **ESC c DELETE** or **ESC c CTRL-D** deletes all the characters on a line from the cursor to the first occurrence of 'c' on the same line; 'c' may be any character. The text to the right of the deletion shifts left. **ESC DELETE** or **ESC CTRL-D** deletes every character to the end of a line from the cursor.

CTRL-^deletes an entire word regardless of the cursor's position within that word. The text to right of the deleted word shifts left. If the cursor is on a space preceding a

word, all spaces from the cursor to the start of the word are deleted. In PPWrap mode, if the word deleted is the only word on a line, a line delete is performed, closing up the created space between the lines above and below.

ESC CTRL-^deletes an entire line regardless of the cursor's position within the line, and all subsequent lines are moved up. **ESC nn CTRL-^**deletes 'nn' lines at and beneath the cursor.

ESC #mm,nn CTRL-Δdeletes lines 'mm' to 'nn', including lines 'mm' and 'nn.' This command also takes several seconds to execute, so the Status Line will show BUSY while the command is being carried out; type-ahead may be used.

A '$' may be used instead of a number for 'nn' to stand for the last line in the file.

Splitting and Joining Lines

	PIE Command Key	Function
ESC	CTRL-E	Splits a line at the cursor
ESC	CTRL-\	Joins two lines together

ESC CTRL-E splits a line at the cursor and moves whatever text was to the right of the cursor to the left margin of the new line.

ESC CTRL-joins a line below the cursor to the end of the line the cursor is on. The cursor must be positioned following the final character in a line, or else the Status Line says "CLEAR TO EOL (end of line) FIRST." If the line below the cursor is too long to be appended to the current line, the message "NOT ENOUGH ROOM ON LINE" is displayed on the Status Line.

Copying and Moving Lines

The ability to copy lines into memory and move them to different positions in your text is a powerful editing function. Using the copy functions, any amount of text may be copied from one position to another position within a file. Information held in the copy buffer remains there until another move or copy command is given, and so may be recalled repeatedly. It even remains intact after loading in a new file so that short blocks of text may be moved between files easily.

	PIE Command Key	Function
	CTRL-L	Copies line at cursor into memory
ESC nn	CTRL-L	Copies 'nn' lines into memory
	CTRL-	Copies line at cursor into memory and deletes it from screen
ESC nn	CTRL-\	Copies 'nn' lines into memory and deletes them from screen
	CTRL-O	Recalls lines from memory and copies them onto the screen at cursor
ESC #mm,nn	CTRL-O	Moves lines 'mm' to 'nn' to current cursor position

CTRL-L copies a single line of text into memory. The Status Line blinks once while the line at the cursor is being copied. **ESC nn CTRL-L** copies 'nn' lines beginning at the cursor into memory, up to a maximum of 21 lines of text, or one full window.

The **CTRL-** command copies a single line of text at the cursor into memory and deletes it from the screen. This is the safest method you can use for deleting a line of text, because it can always be recalled using **CTRL-O**. You can save a copy of up to one window, or 21 lines, of text with **ESC nn CTRL-** which saves 'nn' lines in memory while deleting them from the screen.

The **CTRL-O** command allows you to recall the set of lines placed in memory with either **CTRL-L** (which copied them into memory) or **CTRL-** (which copied and deleted). The text recalled is inserted at the current cursor position. The cursor may be repositioned and the **CTRL-O** command may be used again to insert as many copies of the block of lines as you like.

ESC #mm,nn CTRL-O moves a range of any number of lines. Move the cursor to the place in the text where you want to move lines 'mm' to 'nn.' Enter the **ESC #mm,nn CTRL-O** command, and the range of lines 'mm' to 'nn' will be retrieved from memory and placed at the cursor.

Again, '$' may be used to represent the end of file.

Search and Replace

A "string" is a sequence of characters. A string might be a sentence, a word, or a single character. PIE limits search strings to a maximum of 26 {58 in 80-column versions} characters. The next set of commands instructs PIE to search for a string and to position the cursor at the string when it has been found. Failed searches produce a "SEARCH KEY NOT FOUND" message on the Status Line. (The word "KEY" refers to the search string.)

PIE Command Key		Function
	CTRL-Z	Searches forward for current search string
ESC s	CTRL-Z	Sets search string to 's' and searches forward
	CTRL-Q	Searches backwards for current search string
ESC s	CTRL-Q	Sets search string to 's' and searches backward

To find a string, type **ESC** to display the ARG: prompt on the Status Line. Enter the string ('s') you want found. After the string is entered, press **CTRL-Z** to search toward the end of the file or press **CTRL-Q** to search toward the beginning of the file. PIE Writer remembers the search string until you type in a new one. To search for the next occurrence of the same string, press **CTRL-Z** to search forward again, or **CTRL-Q** to search backward again.

If the string exists but is split across two lines it will not be found.

A search string can contain ambiguous or "wild card" characters (except for the first character). The wild card character matches any letter. The ASCII DEL or rubout character is the wild card, and is entered by typing **CTRL-C 3** or **CTRL-C #**.

For instance, a string consisting of the letter "C," the letter "A," and the DEL character will match "CAT," "CAP," and "CAR" during a search.

	PIE Command Key	Function
	CTRL-X	Replaces next occurrence of search string with replace string
ESC s	CTRL-X	Sets replace string to 's', then replaces next occurrence of search string with 's'
ESC s1 ESC s2	CTRL-X	Sets search string to 's1', replace string to 's2', and replaces next occurrence of 's1' with 's2'
	CTRL-W CTRL-X	Replaces all occurrences of search string with replace string
ESC s1 ESC s2	CTRL-W CTRL-X	Sets search string to 's1', replace string to 's2', and replaces all occurrences of 's1' with 's2'

The search and replace commands make use of two different strings, a 'search' string and a 'replace' string, each of which is stored in a separate place in memory ('s1' refers to the search string and 's2' to the replacement string). As can be seen above, there are several ways in which these two strings may be entered into memory and new values given to the the search and replace commands.

ESC s1 ESC s2 CTRL-X searches forward for a string ('s1') and replaces it with another string ('s2'). Pressing **CTRL-Z** (or **CTRL-Q**) will then search forward (or backwards) to find the next string that matches the search string. Typing **CTRL-X** performs another replacement at the next occurrence of the search string; that is, it does an automatic forward search. To enter a different replacement string but leave the search string intact, use **ESC s CTRL-X**.

If the replacement string ('s2') causes the line to exceed the available space on the line, the command will not execute, and the Status Line will display "NOT ENOUGH ROOM ON LINE". If the search string cannot be found, then the replacement cannot be made, so the Status Line displays "SEARCH KEY NOT FOUND". Remember that

CTRL-X does only a forward search.

To change every occurrence of a string throughout the text, use **ESC s1 ESC s2 CTRL-W CTRL-X**. This looks for the search string ('s1') and replaces it with the replacement string ('s2') everywhere in the file. This function is called a "global" search and replace.

Every occurrence of the search string will be replaced, unless a line is made too long by the replacement string, in which case the command quits. You may edit the line, the press **CTRL-W CTRL-X** again, and the global search and replace continues forward searching and replacing using the same strings.

The sequence **CTRL-W CTRL-X** takes the most recent search and replacement strings and performs a global search and replace from the cursor position to the end of the file.

Note that using **ESC s CTRL-P** to make an insertion changes the current replacement string.

Text Entry Modes

	PIE Command Key	Function
ESC	RETURN	Toggles PPwrap with Manual mode
ESC I	RETURN	Enters Indent mode

A standard typewriter requires a carriage return at the end of every typed line. PIE Writer's Manual mode works the same way: the **RETURN** key must be pressed to advance the cursor to the beginning of the next line. Manual mode is what you are in when you first enter the Text Editor.

ESC RETURN (or **ESC P RETURN**) sets the PPWrap mode from the default Manual mode. PPWrap mode allows you to type continuously without pressing the **RETURN** key: whenever a character is typed beyond the bell column, a blank line is inserted below the current line and all of previous characters in the same word are automatically moved down. The entire word wraps around the screen and begins at the beginning of the new line. A new line is also inserted if **RETURN** is pressed when the cursor is at the end of a line.

ESC I RETURN lets you enter the Indent mode. In this mode, pressing the **RETURN** key moves the cursor to the position below the leftmost charcter on the previous line. This is useful for entering outlines, lists, and program text, such as Pascal or Assembly code.

```
In the Indent mode
    a line indented this much
    is indented again automatically
    after every carriage return.
```

You must press **RETURN** to perform a carriage return in the Indent mode. Text does not wrap around the screen automatically. Pressing **RETURN** does make room for any new text you may want to enter, however, by forcing all text below the cursor down one line.

ESC RETURN from either PPWrap or Indent mode returns you to Manual mode.

Displaying the Help Screen

PIE Command Key		Function
ESC H	RETURN	Displays Help screen

ESC H RETURN displays the Help screen, a listing of all the simple functions and their command keys. However, this command is available only if the Help file already has been loaded from the Command Processor by typing **%H**. (Refer to the section on the Command Processor in this chapter.) Otherwise the message "INVALID PARAMETER" is output.

Using PIE with the Help screen loaded reduces the size of your edit buffer by 1024 characters. It is intended to be used while learning the Text Editor commands.

By pressing **RETURN** while viewing the Help screen, you are returned to the point in the file where you left off.

Cursor-Defined Commands

With cursor-defined commands, you allow the horizontal and vertical motion of the cursor to define the area of the screen you want a command to be effective in, so that the same command can be applied to more than one line of text at a time. Cursor-defined commands can be carried out only for on-screen text below or the right of the cursor's position before the cursor-defined command is entered.

Cursor defined commands allow you to count lines and columns, using cursor movement, to specify the range of a command, just as a numeric value can be used to count those same lines and columns. For example, **ESC 8 CTRL-E** inserts 8 blank lines at the cursor.

The cursor-defined analog of this command is described below.

	PIE Command Key	Function
ESC ⇨⇩	CTRL-E	Block move right
ESC ⇨⇩	CTRL-\	Block move left
ESC ⇨⇩	CTRL-^	Block move left with delete
ESC ⇩	CTRL-L	Copy lines to memory
ESC ⇩	CTRL-\	Delete lines to memory
ESC ⇩	CTRL-^	Delete lines

The symbols ⇩ and ⇨ are used to represent vertical and horizontal cursor movement. Commands which only have the ⇩ symbol only allow vertical movement; addition of the ⇨ means that the command uses horizontal cursor movement as well.

Cursor-defined modifiers are entered by first typing **ESC**, then entering one of the seven simple cursor movement commands noted below. As soon as a cursor command is typed following the **ESC**, the original cursor position is replaced on the screen with the @ symbol, and the message "CURSOR DEFINED" is displayed on the Status Line. For vertical cursor-defined commands, the net cursor movement must be down; upwards cursor movement is allowed, but not above the original cursor position. The cursor movement is used indicate the range over which the command is effective; the original cursor position serves as one end point, the final cursor position as the other.

Subject to this restricition, the simple commands ↓ and ↑ function as usual to move the cursor vertically. The **CTRL-G** cursor move, however, works differently. Instead of toggling between Home and Bottom Home, as a modifier, it toggles between the original cursor line and the bottom line on the screen, remaining in the current column.

The horizontal cursor-defined commands use the →, ←, **TAB**, and **CTRL-A** simple commands to define the horizontal range over which a command is effective. Similarly to the vertical cursor-defined commands, the net cursor movement must be to the left of the original cursor position.

All of the horizontal cursor-defined commands can also be used with vertical cursor movement to define the range of lines over which the horizontal function is to be performed; thus "blocks" of text on the screen may be shifted.

ESC ⇨⇩ CTRL-E performs a cursor-defined block move right. Suppose that the cursor is positioned on the 'T' in the word 'this' below, and you want to shift two lines of text so that they begin eight columns to the right.

```
THIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

Press **ESC** and then press **TAB**. The Status Line displays the message "CURSOR DEFINED".

The cursor's original position when you entered the command is marked with an @ symbol on the screen.

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

Assuming your tab stops are set to 8, the **ESC TAB** tabs the cursor to the 'A' in 'AN':

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

↓ moves the cursor down one line so that the cursor is positioned over the 'S' in 'USE' on the second line:

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

CTRL-E inserts spaces from @ to the current cursor position:

```
          THIS IS AN EXAMPLE OF
          HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

ESC ⇨⇩ CTRL- performs a cursor-defined block move left. The **CTRL-** command is preceded by a simple horizontal command (to indicate the number of spaces you want to shift the text left) and a simple vertical command (to indicate how many lines you want shifted). Note that the cursor movement is in the direction *opposite* that of the text shift. In the example below, you can shift the block of text to the left by first positioning the cursor in the column where you want the first letter of the line to end up, then pressing **ESC**:

```
@    THIS IS ANOTHER EXAMPLE OF
     HOW TO USE HORIZONTAL AND
     VERTICAL CURSOR-DEFINED COMMANDS
```

Move the cursor so that it is positioned over any part of the text in the first line with → or **TAB** commands, then use a ↓ to move the cursor onto the line below it. Type a **CTRL-** command to see the screen below:

```
THIS IS ANOTHER EXAMPLE OF
HOW TO USE HORIZONTAL AND
    VERTICAL CURSOR-DEFINED COMMANDS
```

ESC ⇨⇩ CTRL-^ shifts text to the left, but also deletes characters as defined by the cursor movement. In the example above, the cursor would have had to have been positioned horizontally on the 'T' in 'THIS,' then moved to the 'H' in 'HOW' with a ↓ to get the same result. Positioning the cursor anywhere within the lines would have deleted all the characters to the left of the cursor, then shifted the remaining characters on the lines left to the column where the @ was first set.

ESC ⇩ CTRL-L copies lines into memory as defined by the cursor movement; it functions identically to **ESC nn CTRL-L**, with "nn' simply being defined by the cursor movement rather than with a number. The entire lines are placed in memory regardless of the column the cursor is in.

ESC ⇩ CTRL- places lines into memory as defined by the cursor movement and deletes them from the screen; again, it is similar to **ESC nn CTRL-**

ESC ⇩ CTRL-Δ deletes lines as defined by the cursor movement, and the lines below those deleted close up the space created.

Entering Control Characters

Control characters cannot be entered into a text file directly using the Text Editor for the simple reason that all of the keyboard control characters are interpreted as Text

Editor commands. Naturally, it is desirable to be able to enter any ASCII character into your file. (A typical application would be to imbed printer commands in your file.)

PIE Command Key	Function
CTRL-]	Allows entry of next character as a control code

CTRL-] lets you enter any control character literally, that is, without interpretation. Executing a **CTRL-]** produces the message "ENTER CTRL CHAR" on the Status Line. The control character you enter appears on the screen as a flashing character. {On 80-column versions, it depends on your board, but they are usually displayed in inverse video.}

Exiting the Editor and Transferring to the Command Processor

The final Text Editor command transfers you to the Command Processor (CP).

PIE Command Key	Function
CTRL-@	Exits Editor and returns to Command Processor

You exit all PIE Text Editor files and transfer to the PIE Command Processor by executing a **CTRL-@**. Your text remains intact in the buffer. You can reedit it, but it will not be stored permanently on diskette until you enter a save or write command while in the Command Processor.

PIE COMMAND PROCESSOR

You are transferred to the Command Processor after choosing the first option in the System Menu, PIE TEXT EDITOR; or after exiting the PIE Text Editor with a **CTRL-SHIFT-P** (or **CTRL-@**) command; or when returning to PIE from FORMAT. The Command Processor is used to handle the loading and saving of all PIE Writer files on diskette and to provide additional file-manipulation and input/output capabilities.

The Command Processor has three types of commands: Control, Input/Output, and Auxiliary. All commands must be followed by a **RETURN** and are entered in response to the Command Processor prompt:

```
COMMAND?:
```

Prior to pressing **RETURN**, you may delete an entry by typing **CTRL-X**. For users with an Autostart ROM, the Command Processor also supports cursor moves and screen copying using **ESC I-J-K-M** as described in your the reference manual for your computer.

Control Commands

CP Command	Function
N(EW)	Edit new file; clears current file
E(DIT) n	Re-edit current file (optionally starting at line 'n')

F(ORMAT)	Transfer to FORMAT program
C(ATALOG)	Catalogs current drive
?	Display remembered filename (fn), set with L fn, S fn, <fn,>fn
LE(NGTH)	Display remembered filename, length, and available memory remaining
CTRL-Y	Enter system monitor; **CTRL-Y** returns you to Command Processor
CALL addr	Calls machine language routine at addr; may be signed decimal or $hex

Parentheses around part of a command in the above table indicate that typing the enclosed part is optional.

N (or **NEW**)—Clears the edit buffer before beginning a new text file. The **N(EW)** command is followed by the precautionary prompt OK TO CLEAR?: A **Y** (for yes) response produces the empty text widow; an **N** (for no) redisplays the CP prompt.

E (or **EDIT**) **n**—Returns you to the text editor without clearing the previous contents of the buffer, so that you can begin editing a file or continue inputting text. This command must be used after loading a file from diskette for re-editing.

The cursor appears at the beginning of the text for a recently loaded file; otherwise, the cursor reappears at its previous position if you have worked with this file and have transferred to the Command Processor temporarily (for example, to update the diskette file).

Entering an optional 'n' value allows you to specify the line number of the file you would like the cursor to appear at the beginning of. If the line number is greater than the highest line number in the text buffer file, the cursor appears at the beginning of the file's last line. (The command **E $** will also jump to the end of a file.)

F (or **FORMAT**)—Transfers you to the FORMAT Text Processor, which formats a file for output to the printer, screen, or diskette.

C (or **CATALOG**)—Displays the filenames on the currently active diskette. The slot, drive and volume number of the disk drive whose catalog you want displayed may optionally be specified if more than one drive is being used.

?—Displays the title of the last loaded or saved file. This filename is the 'remembered filename' and may be used as a shorthand notation for that filename in any command, as in the following examples:

- **S ?**—*Saves 'B' type file to diskette with remembered (current) filename. (You may optionally specify a different slot, drive, or volume number.)*
- **S ?.BAK**—*Saves a file with the characters .BAK appended to the remembered filename.*
- **> ?**—*Outputs 'T' (DOS text) type file using the remembered filename.*
- **L ?**—*Loads file. This would generally be a re-load of the file in memory.*

Using the ? token as part of a filename when appending or saving to a 'side file' (as in S ?.BAK) does not change the remembered filename.

LE (or **LENGTH**)—Displays the remembered filename, the number of characters in the file, and the remaining memory available.

CTRL-Y—Transfers you to the machine language monitor through (CALL-151). Another **CTRL-Y** returns you to the CP through the warmstart location, $806. (This feature is of interest only to users who are also programmers.)

CALL addr—Calls the machine language routine at 'addr,' a given signed decimal or hex address. Again, this is of interest to the programmer.

Input/Output Commands

CP Command	Function
L(OAD) filename	Load 'B' type file into edit buffer
S(AVE) filename	Write 'B' type file to diskette
>filename	Write 'T' type file to diskette
(begin,end)>filename	Extract line range and write to 'T' type file
>>filename	Append to 'T' type file
(begin,end)>>filename	Append range to 'T' type file

<filename	Load 'T' type file
(begin,end)<filename	Load range from 'T' type file
<<filename	Append to memory from 'T' type .file
(begin,end)<<filename	Append line range to memory from 'T' type file

Again, parentheses around part of a command in the above table indicate that typing the enclosed part is optional.

L (or **LOAD**) filename—Loads a 'B' type file into the text buffer. All DOS parameters are allowed.

S (or **SAVE**) filename—Saves, or writes, the contents of the edit buffer as a 'B' type file. Again, all DOS parameters are allowed.

'B' type files are DOS Binary type files, and are the best to use for fast loading and saving of documents.

The processing of 'T' (DOS 'Text') type files is also possible from the Command Processor. This type file is slower to read and write but allows more flexibility than the 'B' type files accessed with the Load and Save commands.

The primary commands for 'T' type files are '<' for input (reading) and '>' for output (writing).

It is sometimes desirable to manipulate only part of a file. This option is available when working with 'T' type files; to specify a range of line numbers in a file or in the edit buffer, precede the command with a parenthetical expression as below, where 'begin' means the lowest line number and 'end' means the highest line number.

>filename—Writes a 'T' type file from the text buffer to diskette. If "filename" already exists, its previous contents are replaced.

(begin,end)>filename—Extracts a line range from the buffer and saves it as a 'T' type file.

>>filename—Appends the contents of the text buffer to the end of an existing 'T' type file.

(begin,end)>>filename—Appends a line range from a file to the end of a 'T' type file.

<filename—Reads a 'T' type file into the text buffer, clearing the previous contents of the buffer.

(begin,end)<filename—Reads a line range from a 'T' type into the text buffer.

<<filename—Appends a 'T' type file to the end of text in the text buffer; the existing text in the buffer remains intact.

(begin,end)<<filename—Appends a line range from a 'T' type file to the end of text in the text buffer.

It is possible to prefix line numbers to every line or to a range of lines in a file before writing those lines as a 'T' type file:

#>filename—Saves a 'T' type file to diskette and prefixes each line with its corresponding line number (line numbering is identical to that in the Text Editor). For instance, the sixth line of unformatted text in a file would be saved on diskette with the number 6 preceding it, using this command.

#(begin,end)>filename—Extracts a line range from a file and saves it to a 'T' type file, each line prefixed with its corresponding line number.

The read and write commands ('<' and '>') may also be used with slot numbers in place of 'filename'; in that case the writing or reading is to or from the slot rather than a diskette file. For example, where "slotnumber" is any valid I/O slot (a number in the range 1-7):

>#slotnumber—Sends all the characters in the text buffer to the numbered slot. If no card is installed in the slot, or if the card has no ROM to handle such a function, the system will 'hang'; in this case, press RESET to return to the CP prompt.

<#slotnumber—Brings characters from the specified slot directly into the edit buffer, clearing its previous contents. This function is generally not practical, because the system has no way of knowing when to stop inputting characters and return to the CP; instead, another version of this command is used that allows the specification of an "end of file" character.

(begin,end)<#slotnumber—Inputs a line range from the specified port to the text buffer; a "line" is a sequence of input characters terminated by a carriage return.

Any ASCII character may be designated as the end of file (eof) marker for input or output when using the '<' or '>' commands. The selected ASCII character is represented using either **@xx** (decimal) or **@$xx** (hexadecimal) as a marker in the

following examples, where 'xx' is a valid decimal or hexadecimal number. If CTRL-D is the end of file marker, for example, then it would be represented by @$84 ($84 is the hex code for CTRL-D with the high order bit set, which is normal for Apple and Franklin keyboard characters). The command @$84<#0 will accept input from the keyboard and place the characters in the text buffer until a CTRL-D is entered. This may be a little confusing visually because you cannot see the characters coming into the buffer, and the only signal of a completed sequence is the reappearance of the flashing cursor. If it does not appear when expected, press **RESET** to redisplay the CP prompt.

(eof marker)>#slotnumber—Sends the entire contents of the text buffer to the specified port until the ASCII character in the eof marker is encountered, which causes transmission to end and the CP prompt to reappear.

#(eof marker)>#slotnumber—Sends the contents of the text buffer to a specified port, each line preceded by its corresponding line number, until the end of file character is encountered.

(eof marker)<#slotnumber—Inputs data from the specified port into the text buffer until the eof marker is encountered; overwrites previous contents of the buffer.

(eof marker)<<#slotnumber—Appends the data brought in through the specified port to the end of the text buffer until the eof marker is encountered.

One final form of the read and write commands gives the programmer the ability to write a special input or output 'driver' and access it with the following syntax:

```
>addr

<addr
```

This simply means that if the user has a routine in place at the address addr, (where addr is either a signed decimal or hex address with the $ prefix), the CP will call the specified routine for input or output, as indicated. The routine should return with an 6502 RTS instruction.

Auxiliary Commands

The auxiliary commands are called from the Command Processor by prefixing a '%' (percent mark) to the commands shown below. For the 48K versions of PIE Writer, the PIE Writer system diskette must be in the defined program slot, drive, and volume (the defaults are 6, 1, and 0) to use the auxiliary commands. (This is because the routines that execute these commands must be loaded from the diskette; in the 64K versions, these routines are loaded into the 16K RAM card.)

The first auxiliary command issued while using the CP on a 48K system will cause the disk drive's IN USE lamp to light while the file CP AUX is loaded from the system diskette. Thereafter the full complement of commands below are available from the command processor until EDIT or FORMAT is invoked.

This means that each time you return to the CP and issue a %-prefaced command, the CP AUX file is loaded. Users with 64k systems will find the AUX commands always available without accessing the system disk.

As with other CP commands, all of these commands must end with a **RETURN**.

CP Command	Function
%Q	Quit
%C	Catalog and space on diskette
%H	Load Help file
%FS	File Status
%WC	Word count
%LC	Line count
%B	Backup current file
%M	Return to main menu
%TS filename	Save tab settings
%TL filename	Load tab settings
%CI	Caps inverse
%CN	Caps normal
%CB n	Break lines to length 'n'
%CS n	Slice lines to length 'n'
%CJ	Join broken lines
%LF (-)	Set/reset line feed after RETURN on output

%D filename	Delete file
%L filename	Load file
%U filename	Unlock file
%V filename	Verify file
%R f1,f2	Rename file
%E filename	Exec file
%ST	Start Doubletime
%SP	Stop Doubletime
%RS	Resume Doubletime

%Q—*Quit.* Exits the program to the ']' BASIC level. To "warmstart" without losing the contents of the text buffer, type **CALL 2054** and press **RETURN**. The warmstart will not work after a BASIC program has been loaded.

%C—*Catalog and space on diskette.* Displays the normal catalog information, plus the number of unused sectors remaining on the diskette.

%H—*Load Help file.* Loads the file PIE HELP into memory and enables the Editor command **ESC H RETURN**. Reduces available Editor memory by 1024 characters. This memory can be recovered, and the Help command disabled, by typing **%H-**. The Help screen remains loaded when switching between PIE and FORMAT.

The standard screen display when entering CP from either PIE or FORMAT shows the last valid title, the file length in characters, and the remaining memory in the text buffer. The next three auxiliary commands provide additional information concerning a file's status:

%FS—*File status.* Provides the same text buffer file status information you receive when transferring to the CP from either PIE or FORMAT, plus the file beginning, file end and buffer end addresses in decimal and hexadecimal, the number of lines, and the number of words in the text buffer file.

For example, the %FS command returns the following report on an empty text buffer:

```
FILE:           ?

LENGTH:         0
MEMORY LEFT:    22015

TEXT START:     16384    ($4000)
TEXT END:       16384    ($4000)
BUFFER END:     39200    ($95FF)

# LINES         0
# WORDS         0
```

These last two lines of information can be requested separately:

%WC—*Word count.* Reports on the count of words (character groups separated by spaces) in the text buffer.

%LC—*Line count.* Reports on the count of lines in the text buffer (word groups followed by a RETURN character).

%B—*Backup current file.* Saves the text buffer to diskette with the current remembered filename, after first renaming the diskette file of the same name to ?.BAK, where ? stands for the current filename. If a previous backup file of the same name exists, it will be deleted. (The %B command actually EXEC's the systems text file BACKUP; because of this, the system disk must be on-line when this command is executed even for 64K users.)

%M—*Return to main menu.* Causes the program PIE WRITER: WORD PROCESSING to be run, redisplaying the System Menu. It is used to access PIE Configure or FORMAT Configure.

%TS filename—*Save tab settings.* Allows tab setting to be saved to a diskette file under any name. If your file has complicated tab settings, these can be saved as a separate file:

```
COMMAND?: S FILENAME

COMMAND?: %TS ?.TABS
```

The above sequence saves the contents of the text buffer to the diskette under FILENAME and then subsequently saves the tab settings to FILENAME.TABS.

%TL filename—*Load tab settings.* Allows tab settings to be loaded from a diskette file. The new tab settings replace any default tab settings. A typical sequence would be:

```
COMMAND?: L FILENAME

COMMAND?: %TL ?.TABS
```

This loads the desired file and then loads the tab settings saved previously under FILENAME.TABS.

%CI—*Caps inverse.* Causes capital letters to be displayed as inverse capitals and lowercase is displayed as normal capital letters. This is the default PIE Writer configuration for an unmodified Apple II without a lowercase adapter.

%CN—*Caps normal.* Converts the display to normal upper and lowercase.

%CB n—*Break lines to length 'n'.* The command means "convert and break." It is used when a file with lines of unknown line length has been loaded from diskette, or if the line length is known to be longer than the 64 {128 for 80 column versions}. This can be true in the following circumstances:

- *The file is a BASIC listing in text ('T') file format, with lines frequently exceeding 64 characters {or even 128}.*
- *The file was created using an 80-column board version of PIE Writer and will be edited using a 40-column version.*
- *The file is a data file created by a BASIC program and exceeds 64 {or 128} characters.*

The effect of the %CB command is to shorten any line longer than PIE's screen width, splitting the line between a word and placing the balance on a new line. Whenever a break is forced, a CTRL-Z is inserted in the buffer as a marker. See %CJ below for using this marker.

The %CB may be followed by an optional numeric value. If you wish to shorten a file to a specific maximum line length (this example specifies 50 columns), enter the following:

```
COMMAND?:%CB 50
```

%CJ—*Join broken lines.* The command joins any two lines divided by a CTRL-Z, removing the CTRL-Z while in operation. It restores the possibility of the text buffer

containing line lengths greater than the screen buffer width.

%CS n—*Slice lines to length 'n'.* There are times when lines need to be broken at a given column, but do not need to be rejoined. This break is permanent. %CJ will not restore the previous appearance of a file broken with %CS.

%LF (-)—*Set / Reset line feed after RETURN on Output.* This command instructs the CP to send a line feed after each carriage return to a port when using the text file output command (>). As shipped, the %LF default is set to '-', meaning that no line feed will be sent out after each RETURN unless you enter this command.

All DOS commands, such as DELETE, RENAME, LOCK, and UNLOCK are immediately accessible from the CP. Shorthand versions of several DOS commands are available as auxiliary commands.

All of the disk commands allow the use of the DOS Slot, Drive, and Volume specifications. The ? token for the remembered filename is also allowed using these commands (the ? symbol cannot be used with the normal DOS commands).

%D filename—*Delete file.* To delete any file, use the following form:

```
COMMAND?:%D fILENAME
```

To use the ? token to simply this function, assuming it is the current file which you wish to delete:

```
COMMAND?:%D ?
```

%L filename—*Lock file.* This command prevents you from accidentally destroying essential information by "locking" the file. In order to change the file by re-saving data to it, it must first be unlocked.

%U filename—*Unlock file.* Allows a locked file to be deleted or modified.

%V filename—*Verify file.* This command verifies the internal consistency of a file stored on diskette. If an error is reported, save the file again to a different diskette.

%R f1,f2—*Rename file.* Renames 'f1' as 'f2'. An example using the '?' symbol:

```
COMMAND:%R ?,?.BAK.
```

%E filename—*Exec file.* This is the appropriate way to execute 'shell' commands. See Chapter 6 for a discussion of how to use the shell file command feature.

If you own the Doubletime Printer spooling software (available only for the Apple II and II Plus), the following commands allow you to control background printing directly from PIE Writer. You should install the Doubletime Printer prior to using PIE Writer.

Hayden Software Company does not support or distribute the Doubletime Printer, but PIE Writer includes the following helpful commands.

PIE Configure must be informed that Doubletime has been installed. The file DOUBLETIME PRINTER, which should be transferred onto the PIE Writer system diskette, is loaded the first time the **%ST** command is entered.

%ST—*Start Doubletime.* This command performs the START operation of Doubletime, and begins printing of any files ending with a "/D."

%SP—*Stop Doubletime.* This performs the STOP operation of Doubletime, halting further output without altering spooling "pointers."

%RS—*Resume Doubletime.* This RESTARTs Doubletime for continuation after a STOP. The **%SP %RS** sequence is used to stop printing temporarily and then continue on as before.

FORMAT TEXT PROCESSOR

The FORMAT Text Processor outputs the document you have entered using PIE to the printer, or optionally, the screen or a disk file. It interprets any dot commands within the file and formats the document as you have specified.

There are several output options available from the FORMAT command level. When you enter FORMAT either by selecting option 2 from the System Menu or by transferring there from the PIE CP by typing **F**, you are presented with a series of prompts, as follows:

Prompt	*Option*
OUTPUT TO PRINTER...	A file can be output to the printer, diskette, or screen. To output from 1 to 99 copies of a file to the printer, type the number of copies desired and press **RETURN**. To output a single copy of a file to the printer, diskette, or screen, press **P**,**D**, or **RETURN**, respectively. Formatted output to diskette is for the specialized use of programs such as background printers and is *not* the same as saving the file for further editing later.

PAGE LIMITS...

To print a single page in a file (for example, page 3), type **3,3** and press **RETURN**. To print pages 4 through 12, type **4,12** and press **RETURN**. To print from page 6 to the end of the file, type **6** and press **RETURN**. To stop printing before the beginning of a page, which is desirable when feeding single sheets of paper into the printer, type **S** after you have set your page limits. For instance, to print pages 2 to 5, stopping before each page, type **2,5S**. To stop before printing each page in a file, type **S**.

INPUT FROM DISK...

To get a file from the diskette, type **D**. To get a file still in the text buffer memory, type **M**. To get a catalog of a diskette's files, type **C**. You will be prompted:

ENTER S#,D#...

Enter the drive number holding the diskette whose files you want to catalog (if you are using more than one disk drive). Also, enter the slot number holding the Apple Disk Interface card, if applicable. If neither of these apply to your system, press **RETURN** to catalog the current drive.

FILENAME:

Enter the name of the file you want to work on.

FORMAT DOT COMMANDS

The FORMAT commands allow you to control the layout of your document. The FORMAT Text Processor acts upon the instructions relayed through these commands (called 'dot commands' as explained below), and outputs formatted files to a printer, a diskette, or the screen.

All FORMAT commands are preceded with a period, or dot, hence the name 'dot commands.' Additionally, each command must be typed on a new line with the dot in column one. They may be typed in either upper or lowercase.

Most dot commands can optionally take a number value which modifies the meaning of the command. This value is represented as 'n' in the descriptions that follow. When entering the dot command, 'n' is replaced with the desired value, or you may leave it off. By prefixing 'n' with a '+' or '-,' the current 'n' value is increased or decreased accordingly by the value that follows. The 'n' value may follow the text of the command immediately, or be separated by a space.

The descriptions that follow contain summary tables. The first column of the table is labeled "Command"; it contains the actual format command as it is entered in a PIE Writer file.

Some commands cause the line that follows them to "break," meaning that the FORMAT Text Processor places text following those commands on a new line, instead of continuing to 'fill' the text. The commands that cause text breaks are noted with a check mark (✓) following them.

The next column is "Default". It indicates the value PIE Writer assumes if a command that can take an 'n' value is entered without one.

The "Initial" column indicates the 'n' value that the FORMAT Text Processor assumes for that command even if a dot command is never given to change that value. For instance, if you never specify page length with the **.pl** command, FORMAT assumes that you want a 66-line page.

Both the default and initial values for many of the dot commands may be redefined by selecting the FORMAT Configure option from the System Menu.

The following are the dot commands that FORMAT recognizes and executes.

Paragraphs

The **.pp** and **.np** commands signal the beginning of paragraphs Each activates a set of layout and indentation values when processed by FORMAT. The **.ps**, **.pn**, and **.pi** commands override these default values; they go into effect whenever the next **.pp** or **.np** commands are encountered.

Command	Default	Initial	Function
.pp n ✓	n = +5		Begin paragraph
.ps n	n = 1	1	Modify paragraph .spacing
.pn n	n = 2	2	Lines needed to begin paragraph
.pi n	n = +5	+5	Paragraph indent
.np n	n = 2		Begin no-fill paragraph

A **.pp** command is used to begin a 'fill' mode paragraph and can perhaps best be understood by thinking of it as a shorthand form for a series of simple FORMAT commands:

.sp—There is one line of space between paragraphs.

.ti +5—The first line of each paragraph is temporarily indented 5 additional spaces.

.ne 2—There must be enough room for at least 2 lines of a paragraph that begins at the bottom of a page, or else the entire paragraph begins on the next output page.

.fi—All paragraphs are output in the fill mode.

The **.sp**, **.ti**, **.ne** and **.fi** commands are explained in detail below; the important idea is to understand that the **.pp** command activates several functions, each of which can be controlled independently.

Enter a **.pp** command immediately preceding each paragraph. Following the **.pp** command with a value indents the first line of each paragraph 'n' spaces (or '+n' or '-n' spaces) instead of the default value.

The **.ps**, **.pn**, and **.pi** commands override parameters used by **.pp** (or **.np**, introduced below), and they become active (and remain that way) with the first **.pp** or **.np** command that follows them.

.ps—*Modify paragraph spacing.* This command overrides the .sp default and initial value to increase, decrease, or eliminate space between paragraphs.

.pn—*Lines needed to begin paragraph.* The .pn 'n' overrides the .ne default value (the number of lines required at the bottom of a page in order to begin a paragraph) to increase or decrease the number of lines needed.

.pi—*Paragraph indent.* This command overrides the .ti default and initial value of +5 to alter each paragraph's opening indent.

Once entered, these three commands are carried out every time you enter a **.pp** or **.np** command thereafter. (The **.pi** is ignored by the **.np** command.) You can permanently change the initial **.ps**, **.pn**, and **.pi** values by altering values in FORMAT Configure's Paragraphs and Filling option.

The no-fill paragraph (**.np**) similarly activates three built-in functions.

.sp—There is still one line of space between consecutive paragraphs.

.ne n—The number of lines needed to the bottom of the page before beginning a paragraph on a new page equals 'n.'

.nf—All text is entered in the no-fill mode until a .pp or .fi is encountered.

No-fill paragraphs are the recommended way to prepare output of material that you want printed exactly as it appears on the PIE screen; i.e., tabular material. The default .np value for 'n' is 2. Unlike the .pp command, this value effects the paragraph 'need' value rather than the indent (there is no indent for a no-fill paragraph). This means that two lines are needed at the bottom of a page or else the no-fill paragraph will not be started until the top of the next page.

Layout

The layout dot commands control text filling and adjusting modes, vertical spacing, and page control.

Command	Default	Initial	Function
.fi ✓		yes	Fill
.nf ✓		no	No-fill
.ad ✓		no	Adjust right
.na ✓		yes	No adjust right
.br ✓			Break in filling
.ls n ✓	n = 1	1	Line spacing
.sp n ✓	n = 1		Space down 'n' lines
.bl n	n = 1		Force 'n' lines
.pl n	n = 65	66	Page length
.bp n ✓	n = + 1	1	Begin page
.po n	n = 1		Lines needed to bottom

The default configuration for FORMAT is for the fill mode to be set, so if neither the **.fi** or **.nf** commands are inserted, **.fi** will be assumed.

.fi—*Fill.* In this mode, FORMAT gathers as many words as it needs to fill a line of output text. This mode is used for most normal document preparation, allowing you to enter text continuously in the PIE editor without worrying about line lengths. FORMAT will reorganize the text into lines of the length of you specify. FORMAT 'fills' up lines by gathering words from the input file, and joining them with the line above, until either the line length is exceeded or it encounters either a break command (.br) or any command that causes a break (indicated by the '✓'). A blank input line or a line beginning with a space also causes a break.

Fill mode will automatically turns a series of spaces into a single space, except in two cases. If you type two or more spaces after a period, colon, question mark, or exclamation mark, or if one of these characters is the last in a line, then it is treated as the end of a sentence, and at two spaces may be output.

There is also a way to insert 'unpaddable' spaces that are not eliminated in the fill mode (see the section on 'Character Translation and Definition' near the end of this chapter).

.nf—*No-fill.* When working on a document in which output lines should be broken exactly the way you enter them in PIE, the no-fill mode is the setting to use. It does no rearranging and ignores the current line length. Precede text that should be formatted as entered, such as outlines or tables, with a .nf command (or use the .np command.)

.br—*Break in filling.* In fill mode, the .br command tells FORMAT to start subsequent text on a new output line. In no-fill mode, .br has no effect.

Justification commands allow you to select either right justification or ragged right text. The default is ragged right.

.ad—*Adjust right.* In fill mode, this command begins the right justification of text that immediately follows it, according to the specified line length. If your printer is capable of incremental spacing, it does that as well, provided you have specified the appropriate printer in type in FORMAT Configure.

.na—*No adjust right.* This command cancels right-margin justification and produces a ragged right margin.

Vertical line spacing is controlled with the **.ls**, **.sp**, and **.bl** commands:

.ls—*Line spacing.* This command sets the spacing between lines, putting one less that 'n' blank lines between each line (for instance, .ls 1 gives single spacing, .ls 2 gives double spacing, and so on). The default is single-spaced.

.sp—*Space down 'n' lines.* This command inserts 'n' lines in the document. If there are less than 'n' blank lines left on the page, only the number of line spaces that will fit will be output.

.bl—*Force 'n' blank lines.* You can insert blank lines in your document with the .bl command also. The only difference between the .bl and .sp commands is that the .bl forces 'n' blank lines even if FORMAT must break to a new page to insert all the blank lines; this insures 'n' contiguous lines.

Additional page layout values are controlled with the **.pl**, **.bp**, **.po**, and **.ne** dot commands. The **.pl** and **.po** parameters can be redefined using FORMAT Configure.

.pl—*Page lenth.* This command gives FORMAT the page size, up to a maximum length of 250 lines. The default value for 'n' is 66, which assumes a standard page depth of 11 inches and a standard 6 lines per inch.

.bp—*Begin page.* The .bp command tells FORMAT to begin the text that immediately follows it on a new line at the beginning of a new page.

The initial value for 'n' is 1. Used in conjunction with the head (.he) and foot (.fo) title and '%' page numbering capability, however, you can also use it to begin page numbering on a page other than one. For instance, to insert numbers at the top right-hand corner of a page, beginning with the twelfth page, enter a .bp 12 to begin a new page and set the page number to 12, followed by a .he ''% command.

.po—*Page offset.* The .po command allows you to set up the left-hand margin of your documents for your particular printer. The page offset is the area (in characters) at the left side of the page that the printer's print head must be advanced beyond to begin the left-hand margin. (Alternately, some printers allow manual adjustment of the print head's 'home' position.) The default value for 'n' is 0.

.ne—*Lines needed to bottom.* If there is not enough room to fit at least 'n' lines of text at the bottom of a page, the text that follows is forced to begin on a new page. The default for 'n' is 1.

Since the .ne command does not cause a break in fill mode, you may want to precede it with a .br command. This places all previous text at the bottom of a page before the .ne command checks to see if enough lines are available to output the text

that follows it.

Indentation

Elements that may require special left or right indents, such as extracts, outlines, or lists, can be formatted with the **.ll**, **.in**, and **.ti** commands. The initial setting for the **.ll** command can be altered by reconfiguring a selection in FORMAT Configure's Page Format option.

Command	Default	Initial	Function
.ll n	n = 65	65	Line length
.in n ✓	n = 0	0	Indent
.ti n ✓	n = 0	0	Temporary indent for text line

Plus and minus values can increase or decrease a current value for many FORMAT dot command that takes an 'n' value. An indenting command without a plus or minus prefix to the 'n' value sets the indent to an absolute column. For instance, a **.in 5** command sets the indent 5 spaces to the right of the left margin; a **.in +5** command sets the indent 5 spaces to the right of the current indent.

.ll—*Line length.* The .ll command allows you to set the length of each output line of text, up to a maximum of 132 characters. The .ll command can be used to indent the right-hand side of a page. For instance, to indent an extract of quoted material 5 characters from the left and right margin, precede the quoted material with .in +5 to indent the left side 5 spaces .ll -5 to indent the right side 5 spaces (by shortening the line length).

.in—*Indent.* This command indents all text that follows it 'n' character spaces from the left margin. Plus or minus values are valid 'n' variables, so outlines and lists, elements that can have constantly changing indents, are best formatted using the .in command.

.ti—*Temporary indent for next line.* This indents the next line of text, then returns subsequent lines to their previous indent. This can be used, for instance, to outdent the number of each item in a numbered list, yet still left-justify the items themselves. A plus 'n' value allows you to indent a line of text 'n' additional character spaces to the right, then return the alignment of all subsequent lines to the previous indent.

Running Titles and Margins

The **.he** and **.fo** commands allow you to create running titles that appear at the top or bottom of every page of your document; the **.tl** command lets you place a single title anywhere in your document. These commands are most useful when specifying running heads or the consecutive numbering of pages.

Command	Default	Initial	Function
.he t	t = ''''	''''	Head title
.fo t	t = ''''	''''	Foot title
.tl t	t = ''''	''''	Flush left, center, flush right

.he t—*Head title.* Text that follows the .he command on the same line is output at the top of every page of your document. You can change the head title with .he command anywhere on the page preceding the one on which you want it to appear. The top margin defaults (see the .ml and .m2 commands below) print the head title 1 line from the top of the page and 2 lines above the text. Since the head title is printed after .m1, it may be moved by adjusting this margin.

.fo t—*Foot title.* Prints a title at the foot of every page of your document. The title to be printed follows the .fo command on the same line. The foot title can be altered anywhere before you want it to appear. The bottom margin defaults print the foot title 1 line below the text and 2 lines from the bottom of every page. Since the foot title is printed after .m3, it may be moved by adjusting this margin.

.tl t—*Flush left, center, flush right.* The title following the .tl command on the same line is printed once only, exactly where it is entered in the text. The .tl command could be used to right-justify the date in a letter or to center a subhead.

Unlike the .ce command, the .tl does not cause a break. If no title 't' is specified, the .tl command outputs a blank line without causing a break.

The pattern for entering all three title commands is the same: up to three "parts" of a running title, separated by any limiting character (any character that does not appear in any of the parts), can be entered. Part 1 begins at the left margin; Part 2 is centered; and Part 3 is flush right.

To set all three parts in a running title, simply separate them with a valid delimiter (such as an apostrophe, in the example **.fo 'Part1'Part 2'Part3'**). To omit any part of the running title, simply omit that part, leaving the delimiters intact (for instance, **.he**

''**Part 3**' prints only "Part 3" flush right at the top of the page). You need not insert a delimiter to the right of the last section you want printed. (For example, **.tl ''Part 1** is a correct command for inserting "Part 1" flush left on the next line on the page).

You can instruct FORMAT to page your document for you automatically by using the **.he** or **.fo** command with the percent sign. For instance, to have the page number appear in the top right-hand corner of every page of your document, beginning with page one, you would enter the **.he ''%** command before entering any page one text. To begin paging a document anywhere after the first page, you must set the page number with a **.bp n** command, which specifies that paging should begin on the next page, page 'n'. The percent sign in the title command page numbers a document at the place where it appears in a head or foot title, beginning with page 'n' (which defaults to 1), and incrementing its value by one as each page is printed.

The maximum line length for **.he** and **.fo** titles is the text line length at the time the **.he** or **.fo** title is specified.

Head and foot titles are unaffected by indents or temporary indents. However, either of the first two parts of a head or foot title can be shifted right by inserting spaces between the delimiter and the beginning of the text part.

The margin commands allow you to change the top and bottom margins and the relative positions of head and foot titles. The initial values can be changed permanently by altering selections in FORMAT Configure's Top and Bottom Margins option.

Command	Default	Initial	Function
.m1 n	n = 2	2	First top margin (to and including head title)
.m2 n	n = 2	2	Second top margin
.m3 n	n = 1	1	First bottom margin (to foot title)
.m4 n	n = 3	3	Second bottom margin

.m1—*First top margin (to and including head title).* The 'n' value that you enter with the .m1 command determines the number of lines from the top of the page down to, and including, the head title. If you insert a head title without a .m1 command, the head title will be output one line from the top of your page. Since the .m1 value includes space for the head title, setting .m1 to 0 "turns off" the header.

.m2—*Second top margin.* This 'n' value specifies the number of lines between the head title and the first line of text, excluding the first line of text.

.m3—*First bottom margin (to foot title).* This margin dictates the number of lines between the last line in the main body of your text and the foot title, excluding the foot title.

.m4—*Second bottom margin.* The fourth margin tells FORMAT how many lines of space there should be between the bottom of the overall page, up to, and including, the foot title. Just as .m1 0 turns the head title off, .m4 0 turns the foot title off.

If none of the four margin commands are entered anywhere in a file and no head or foot titles are specified, FORMAT prints your document with four lines of space at the top and bottom of every page.

Centering, Underlining and Boldface

PIE Writer is capable of centering, boldfacing, underlining, as well as capitalizing text anywhere on your page. Single words can be emphasized in the fill mode by placing them on their own lines preceded by the dot command on the line above.

Use FORMAT Configure's Printer Interface option to set up the underlining and boldfacing techniques that fit your printer.

Command	Default	Initial	Function
.ce n ✓	n = 1	0	Center
.bf n	n = 1	0	Boldface
.ul n	n = 1	0	Underline
.cp n	n = 1	0	Capitalize

.ce—*Center.* This command centers the following 'n' lines of unformatted text. Each line is centered individually, whether the command is entered while in the fill or no-fill mode.

.bf—*Boldface.* The .bf command restrikes text or changes into a boldface type font for text (depending on your printer) on the following 'n' lines of text.

.ul—*Underline.* Causes the next 'n' lines of text to be underlined. Depending on the type of printer and the configuration selected, underlining is implemented either by

printing an underscore, a backspace, and then the character to be underlined; or by a start/stop sequence turning underlining on and off at the printer; or by printing the entire line, executing a carriage return without a line feed, and re-printing a line of underscores.

The underline command will only cause alphanumeric characters to be underlined; spaces and punctuation marks are not underlined.

.cp—*Capitalize.* You can capitalize 'n' lines of text by immediately preceding them with the .cp command.

File Switching and Form Letters

The following set of FORMAT dot commands implement file switching—that is, the selection of another file as the source of FORMAT's input—useful for easy form letter production or when a single document, such as a long report, is too large to fit into the text buffer all at once.

Command	Default	Initial	Function
.nx filename	prompts		Chain to next file
.op filename	prompts		Open data (secondary) file
.rb -	space		Read block ('-' optional)
.sb n	n = 1		Skip blocks
.ng			Skip blocks to end of group

The text buffer has enough room for approximately five and a half pages of text (assuming 65 characters per line of text, 60 lines of text per single spaced page). Any document that exceeds that length will not fit in the text buffer. The **.nx** command allows you to string together files for continuous document output.

Each 'filename' specified in a **.nx** or **.op** command is converted to uppercase.

.nx filename—*Chain to next file.* Insert the .nx command at the end of any file to chain to another file (named "filename"). When the FORMAT Text Processor has finished formatting the first file, it loads the "filename" automatically (or if a filename

is not specified it sends a prompt to the screen asking for the file's name). The second file 'filename' will be output immediately following the first file, as if the text from both files were actually only one file. Many files (an entire book manuscript, for example) can be chained together in this way.

If no file of the name 'filename' is stored on the diskette, FORMAT outputs the first file until it reaches the .nx command, then interrupts the output process and displays FILE NOT FOUND.

When FORMAT encounters the first .nx command, the current contents of the text buffer are saved in a file named TEMP*** before writing over the text in the buffer with the second file. The original contents are restored in the buffer when the entire document has been output.

All commands and text that follow the .nx command in the first file are ignored.

The **.op** command allows a primary file to access a secondary data file (for instance, containing a mailing list) and to merge "blocks" from the secondary file into the primary document with **.rb**, **.sb**, and **.ng** commands. Using these four commands in your primary file allows you to produce accurate form letters easily.

.op filename—*Open data (secondary) file.* Normally the data file contains names and addresses, phone numbers, and any other information that you would want to manipulate for insertion in a form letter. The .op file can contain formatting commands as well, except for any of the commands that access it (.op, .rb, .sb, and .ng). A .nx command may be inserted at the end of a secondary file to chain together a long mailing list, for instance.

While the .op command can be inserted anywhere prior to the block manipulation commands in your primary file (the file containing the stable, main body of your form letter), it is best to place it at the beginning.

If there is no file named 'filename' existing on diskette, the FILE NOT FOUND message is displayed after output is interrupted when FORMAT encounters the valid .op command.

Two secondary data files cannot be opened simultaneously from the same primary file. The second .op command encountered in a form letter is ignored.

Entering the **.op** command in the primary file does not access a secondary data file, it only opens it. To read any information from a secondary file, the **.rb** command must appear in the primary file anywhere after the **.op** command. The **.rb** command should appear, however, at the exact point in the primary file where data from the secondary file, such as an address, must be inserted.

.rb-—*Read block ('-' optional).* This command instructs FORMAT to read a block of data from the file opened with the .op filename command. Data blocks are defined by the block terminating character, which follows the last line of information

in a block. The initial block terminating character is <, but this can be replaced permanently with any character you choose by changing the fifth value in FORMAT Configure's-Lowercase and Special Character option, or temporarily with the .bc command.

For instance, to insert an address from a data file named ADDRESS as the inside heading in the primary file, the primary file might look like the one below:

```
.OP ADDRESS
.NF
TODAY'S DATE
.SP
.RB
.SP
.FI
DEAR
.RB -
,
```

The data file ADDRESS would look like this:

```
MS. MARKETING DIRECTOR
HAYDEN SOFTWARE COMPANY
600 SUFFOLK STREET
LOWELL, MASS.  01853<
MS. DIRECTOR<
```

The execution of the above commands and text would insert the address where the first **.rb** command is given, providing an inside heading to a typical form letter, and the name where the second **.rb** command is given. The '-' following the second **.rb** tells FORMAT that in fill mode the characters on the next line should be appended to the character preceding the '<' in the data file. In this case, the comma immediately follows "MS. DIRECTOR".

Obviously, data files are much more useful if they contain several blocks of information rather than just one. For instance, a data file might reasonably contain not only a company's name and address, but its phone number, telex number, and so on. Defining each of these pieces of information as a block, with a block terminating character, would allow you to use several **.rb** commands in your primary file and insert blocks throughout a form letter.

Data blocks in a secondary file can be skipped with the **.sb** command. This command is most useful when compiling a form letter that requires only certain blocks of information in a multipurpose data file.

.sb n—*Skip blocks.* Any number of blocks, specified by 'n,' in a data file can be skipped during the output of a primary file with the .sb command. Any commands (including the .nx command) in the skipped blocks will not be executed.

One typical application for the .sb command would be in reading blocks for a form letter from a data file also used to print mailing labels. The primary file might look like this:

```
.OP ADDRESS
.NF
TODAY'S DATE
.SP
.RB
.RB
.SB
.SP
.FI
DEAR
.RB-
,
```

The data file named ADDRESS might appear as shown below:

```
MS. MARKETING DIRECTOR<
HAYDEN SOFTWARE COMPANY
600 SUFFOLK STREET
LOWELL, MASS.  01853<
.SP
ATTN; MARKETING DIRECTOR<
MS. DIRECTOR<
```

Executing these two files would insert the marketing director's name and the company name and address in the appropriate position in the primary file form letter, ignore the information applicable only to printing mailing labels, then pick up the name and comma for the salutation.

Outputting the primary file via FORMAT produces as many form letters as there are "groups" of data blocks, then ends transmission. In other words, a data file with ten groups of five data blocks provides fifty inserts, enough material for ten letters. Blocks in the data file are read or skipped every time a **.rb** or **.sb** is encountered until the data file is exhausted.

Each company name in a data file, for instance, along with an address block, a mailing direction block, a phone number block, and a telex number block, constitutes one "group" of blocks. In the event that you mistakenly form a data file with nine

groups of five data blocks and one group of six blocks, PIE Writer provides a command that synchronizes each group of blocks.

.ng—*Skip blocks to end of group.* Just as the < character signals the end of a data block in a data file, << signals the end of a group of data blocks. Insert the .ng command on the last line of your primary file to make FORMAT search for the << signal following the last line of a group of data blocks. Then insert the << signal at the end of each group of data blocks.

If FORMAT has not encountered the << signal by the time the primary file output is completed, FORMAT ignores any data blocks that have not been read (.rb) or skipped (.sb) by the primary file and advances to the next group for doing the next form letter. This provides a synchronizing feature to overcome occasional typing errors (when one group is longer or shorter than the others) and limiting erroneous output to only a single label, letter, or document.

See Chapter 6 for a more complete discussion of PIE Writer's form letter capabilities.

Interactive Input and Control

A second method for inserting material in primary files is via the terminal communication commands, which are also used to halt output temporarily and to send messages to the screen.

Command	Default	Initial	Function
.gl str	str = '?'	Inserts line entered at keyboard	
.st str	str = ''	Stop (any key continues)	
.tm str	str = ''	Output message to terminal	
.sn	no	Stop next page	

.gl—*Inserts line entered at keyboard.* The .gl command is used to insert text or any command (except .op, .rb, .sb, and .ng) into the file that is being output through FORMAT.

When FORMAT encounters a .gl command, output is interrupted, and the string (str) that follows the command is output to the screen to remind you of what needs to

be input. (If no string accompanies the .gl, the screen displays a question mark.) Respond to the display with text or a command entry, then press RETURN. That entry is then inserted at the point in the file where the .gl command apprears.

In other words, the .gl command gives you the power to create a prompt asking for a specific response. Your response to the prompt is inserted in the file, which is then formatted and output.

For instance, you can use the .gl to obtain often-changed text from the keyboard. If your primary file contains this text:

```
.FI
DEAR
.GL ENTER NAME AND COMMA
.SP
BODY OF LETTER HERE.
```

The .gl then will insert the name and comma you enter after "Dear". (If the no-fill mode were set, your entry would appear at the beginning of the next line.)

Lines also can be joined together when a]followed by a **RETURN** is inserted at the end of a line. This can be particulary useful when obtaining command variables from the keyboard. If your primary or data file contains the command sequence below:

```
.LS ]
.GL LINE SPACING?
```

A response of 2 to the prompt will have the same effect as if a .ls 2 command were given in the original file.

.st—*Stop (any key continues).* This command interrupts the formatted output of a file and displays whatever string (str) follows the .st on the same line. The .st does not allow interactive prompt-and-response, as does the .gl command, but only interrupts output until you press any key.

One useful application for the .st command would be to provide a pause in output for making a physical adjustment to your printer. The .st command does not cause a break, so in fill mode, the interruption of output occurs only at the end of a filled line. Used in conjunction with .br command, though, the .st could be used as shown below:

```
.BR
.ST CHANGE TO RED INK
```

Additional information about this use of the .st command appears in the User Input section in Chapter 6. A more precise means of performing the same function is provided with the .tm command and the]+ sequnce.

.tm—*Output message to terminal.* The .tm command performs the same function as the .st command, except that it does not interrupt the formatted output of a file. The string (str) accompanying the .tm is displayed on the screen whether your file is output to the printer, diskette, or screen.

Used with the]+ signal, the .tm command can interrupt transmission at a precise point in a file and display the accompanying message, allowing you to make physical printer adjustments that produce accurate changes, for instance, with a line of text. See the section on Character Translation and Definition below for additional information on escape characters.

.sn—*Stop next page.* The .sn command performs a temporary stop at the end of the output page it appears on, whether it is inserted in a primary or data file. A blinking cursor appears after transmission is interrupted. Press any key to resume output.

Character Translation and Definition

The character translation commands, **.*** and **.li**, let you enter remarks and dot commands in the PIE Text Editor that are not formatted when a file is output. The character defintion commands, **.ec** and **.bc**, let you enter special characters for FORMAT, printer, or data file merging instructions.

Command	Default	Initial	Function
.*			Comment line (not output)
.ec c	off	']'	Escape character
.bc c	= RETURN	'<'	Block terminating character
.li n	n = 1	0	Literal lines follow

.*—*Comment line.* Preceding any text message with the .* command allows you to enter comments in the PIE Text Editor that are not output during formatting. This can be particularly useful for leaving yourself editing notes, while ensuring that the notes will not be seen on printouts.

For example, you might wish to insert notes in an invoice file, such as:

```
.* INSERT DATE LINE 10: CURSOR # LINE 14
```

or

```
.* TABBING BASED ON $000.00
```

The comment command is useful also when you want to insert source references in documents. This allows you to keep a record of all your sources on diskette, but to leave them out of the formatted document itself. For example:

```
.* REF. PERSONAL COMPUTING, JANUARY 1983
.* ''WORD PROCESSING SOFTWARE,'' VOL. VII, NO.1
.* .PP. 25-30
```

Print any files containing .* commands via the PIE Command Processor for a listing of your document with comments.

.ec—*Escape character.* The escape character is used to notify FORMAT that the next character or characters immediately following are not to be processed in the normal manner.

You can change the escape character default from] (entered with a SHIFT-M on the Apple II Plus in the upper case mode) to any other character you want simply be preceding that other character with a .ec command.

Normally, the escape character disappears when formatted. To cause it to print, enter it twice (for example,]]). If you type .ec without specifying a character, all escape sequences are disabled.

The escape sequences are shown below:

'] '—*An unpaddable (mandatory) space, not to be adjusted.* Guarantees one, and only one, space. (Two in a row guarantees two spaces.) Words separated by unpaddable spaces are never split across, but are kept on the same line together.

The escape character ( ]) is not printed.

'] > '—*A paragraph tab.* Tabs from temporarily "outdented" paragraphs or "side heads" forward to the current indent position by inserting unpaddable spaces. The left alignment of the text to which the side head refers is maintained. For example, this input:

```
.in +15
.pi -15
.pp
a list item]>Paragraph starts here,
after the paragraph tab.  When
formatted, it will be within the
indent specified: the list item
will be ''outdented' at the
negative paragraph indent.
```

Creates this when formatted:

```
a list item        Paragraph starts here,
                   after the paragraph tab.
                   When formatted, it will
                   be within the indent
                   specified: the list item
                   will be 'outdented' at
                   the negative paragraph
                   indent.
```

'] RETURN'—*Join next input line.* Allows lines longer than text buffer width. The first character on the next line (and the characters that follow it) immediately follow the character that precedes the] RETURN. No spaces are inserted between the two characters, so if they are separate words, insert a space before typing the] RETURN.

With this feature, you can join several lines in a row by appending a] RETURN to each line. This allows you to create very wide tables in the no-fill mode. Connected lines must not total more than 132 characters; any characters that exceed this maximum are cut off.

'] "chars" '—*Send all chars with 0 width.* Used for sending printing characters as part of a printer command string without affecting justification. (For example, to get double strike emphasized using an Epson printer, enter **]"<esc> E <esc> G"**. The <esc> stands for the ASCII ESC character, entered in PIE using the CTRL-SHIFT-M (Apple //e and Franklin: CTRL-]) command.)

'] + '—*Stop formatting at once.* Waits for any key to be pressed to continue. The **]+** sequence can be used to stop the printer and change daisy wheels, thimbles, or fonts before continuing with a new typeface.

.bc—*Block terminating character.* You can change the block terminating character from < to any character you wish with the .bc command. If no variable follows the .bc command then each line of your data file will be treated as a block, and a blank line will be treated as the end of a group.

If you change the block terminating character, the group terminating character becomes two consecutive block terminating characters. For instance, if you change the block terminating character to $ then the group terminating character becomes $$.

.li—*Literal lines follow.* The .li command lets you output 'n' lines that begins with a period in a file, without processing them as commands. Dot commands can be entered and printed literally via FORMAT when preceded by a .li command.

ERRORS AND ERROR MESSAGES

When you command PIE Writer to carry out a confusing or incorrect instruction, a system error occurs. PIE Writer will recover from such errors with a minimum of difficulty. Below are examples of situations in which errors occur, along with a description of error messages that may be displayed and the steps that should be taken to recover.

Reset Error

The **RESET** key on the original Apple II can be a source of great inconvenience. When pressed, it causes the computer to stop whatever it is doing.

If you have one of the older Apples with the old monitor ROM, pressing **RESET** while in Text Editor sends you to the system monitor, indicated by the * prompt that appears on the screen to the left of the cursor. To recover your text and continue editing, type **809G**. The screen will be restored exactly as it was when you left off.

For the majority of Apple owners who have the autostart monitor ROM, the **RESET** key has been effectively disabled, so pressing it will not cause any problem.

Recovering Text After Rebooting

There are times when, due perhaps to a hardware malfunction, the only way you can regain control of the computer is by rebooting the diskette. To recover text in memory (assuming that the power has NOT been turned off):

1. *Select option* **5** *(EXIT) from the System Menu.*
2. *When you receive the*] *prompt, type* **BLOAD PIE**, *then press* **RETURN**.
3. *When you receive the*] *prompt again, type* **CALL 2054** *to "warm start" PIE with the text buffer intact, then press* **RETURN**.

It is recommended that you save the text in memory on a diskette immediately.

Disk Errors

Disk errors are generated by the DOS 3.3 operating system. These errors are trapped by PIE Writer and then reported to the user in messages, such as FILE NOT FOUND and DISK I/O ERROR. (A full description of the nature of these errors in found in the DOS 3.3 reference manual.) Generally, by respecifying the command correctly, you can correct the error.

A DISK I/O ERROR, however, may mean that your diskette has been damaged beyond recovery (or it simply may mean that your disk drive is open). This is the main

reason why making backup diskettes is so important.

The FILE NOT FOUND error can occur after entering an AUX command, typing **F** in the Command Processor to transfer to the Text Processor, or trying to return to PIE from FORMAT. This error is displayed if you remove your system diskette from the drive specified as the program drive (normally slot 6, drive 1). Simply reinsert the system diskette and repeat the command.

Another commonly encountered disk error is FILE TYPE MISMATCH. This message appears on the screen if you try to update a file that is already on the diskette using, for example, the **S** (save) command after using the > (write) command (or vice versa) to store it on diskette. This causes an error because those two commands create different type files.

You can always update a file using the same filename—the old file of the same name simply will be overwritten—but the old and new files must be of the same file type. The **S** command and the > command create DOS binary and text type files, respectively. (For the same reason, a file saved using **S** must be loaded using **L**, not <.)

6

Advanced Topics

When used to their full capabilities, the editing, layout, and file manipulation commands can make PIE Writer perform powerful and sophisticated word processing functions. The Mail Merge and User Input features, introduced in the previous section, make printing any form letter an easily managed task if the primary and data files are set up attentively. Shell File structures allow you to define a frequently used Command Processor sequence and perform the consecutive functions with much less typing effort.

Instructions on how to modify an Apple II or II Plus so that the **SHIFT** key provides access to uppercase character entry, as it does on a standard typewriter, are given.

Information on how to transmit PIE Writer files via modem or from computer to computer are also included in the Telecommunication section.

Finally, for the programmer who wants to expand PIE Writer's capabilities even further, the Address Tables section provides PIE Writer entry point and parameter addresses.

MAIL MERGE

Mail Merge allows you to establish data files that can be merged into a sequence of documents. Each document output is identical except where the data in the data file changes.

For instance, if you needed to reach a list of clients to announce a change in policy or price, it would be nice to write a single letter and personalize that letter with data

from a file of names, addresses, and other information.

FORMAT performs this task easily as long as the primary and data files are created thoughtfully.

Primary File

Below is a listing of a primary file, with comments, that utilize Mail Merge's best features. Formatting commands are shown in uppercase for illustration only. They may be entered in either upper or lowercase.

```
.* (File named SALES LETTER)
.*         .* lines are comment lines
.*
.* Open file CUSTOMER LIST, a list of people
.*   to receive the sales letter
.OP CUSTOMER LIST,S6,D2
.*
.* Center 3 lines and capitalize company
.*   name
.CE 3
.CP
Hayden Software Company
600 Suffolk Street
Lowell, MA  01853
.* Turn on no-fill for business address
.*   alignment
.NF
.SP
.* Read store manager's name
.RB
.SP
.* Read store name and address
.RB
.* .PP 0 starts indent 0, and fills so that
.*   ''Dear'', ''name'', and '','' are on 1 line
.PP0
.* Read in salutation name and
.*   place it in between ''Dear'' and '',''
Dear
.RB -
,
.* .AD sets right justification
```

```
.AD
.* .PP starts the first paragraph
.PP
Hayden Software Company is pleased to
announce the immediate availabililty of the
.* underline and boldface title
.UL
.BF
PIE Writer Word Processing System
for the Apple II, II Plus, Apple //e, and
Franklin Ace.
.PP
This powerful addition to the Hayden line of
software will provide your business and
professional users with the most flexible
editing and word processing system
available.
.PP
No hardware boards or adapters are required
for operation of PIE Writer, yet most of the
commercially sold peripherals can be
switched in under software control.
This includes upper and lowercase adapters,
80-column boards, and 16K memory cards.
.PP
PIE Writer works beautifully with standard
Apple II disk drives and  equally well with
Corvus and other hard disk systems.
.PP
PIE Writer has the power and flexibility to
provide for the needs of the beginner and
the expert.
As your skill and understanding grow, PIE
Writer grows with you.
.PP
All of this is provided at a cost less than
that of comparable word processors, which
typically
.BF
require
a Z-80 CPU card and an 80-column video
display board.
```

```
This is a real saving to your customer.
If the user owns a typical 80-column board,
PIE Writer supports it.
.PP
.* Repeat the manager's name
.RB -
, we would like to answer all of your
questions.
Please call or write for more information.
.SP
.NP 6
.IN 33
Cordially,
.SP4
Janice Doe
Sales Manager
.* the .NG will skip to tne next group
.*   for the next letter
.NG
```

You would enter the above document with PIE and save a copy to the diskette under the name SALES LETTER.

Data File

Now take a look at four sample entries in a file called CUSTOMER LIST. This is a data file of manager's names, outlet names, and addresses of stores that may be interested in selling PIE Writer. Notice that the data file's name was specified in the very first line of the primary file by the **.op** command.

To add more names and addresses, begin after the last <<, place block terminators (<) at the end of each line or block of lines, and place a group terminator (<<) at the end of each data group. Then you may place any comments about the store relative to potential sales or problems in the area between the last block terminator and the group terminator. These comments never appear in the printed document.

```
.* (File named CUSTOMER LIST)
.*
.* Each < is a block terminator
.*
.* Each .RB in the primary file reads
.* everything to the next block
```

```
.* terminator and merges it into the file
.*
.* Group 1
Mr. Timothy Gilbert<
Computer Store of Rome
77 Main Street
Rome, N.Y. 99999<
Tim<
Tim<
<<
.* Group 2
Mrs. Helen Drysdale<
Computer Store of Paris
777 Prospect Drive
Paris, TX 11111<
Helen<
Helen<
<<
.* Group 3
Mr. Carl Feinberg<
Computer Store of Moscow
777 Exterior Street
Moscow, PA 55555<
Carl<
Carl<
.* any other information will be skipped .NG
Phone; (415) 777-7777<
<<
.* Group 4
Mr. Sandy Phillips<
Byte Shop
100 Decatur Road
New London, CT 33333<
Sandy<
Sandy<
<<
```

Formatted Output

When the file SALES LETTER is loaded and formatted, four copies of the letter are printed out, each differing only in the name of the store manager and the store address.

All of the comment information from the phone number to the group terminating character is left out of the printed form letter.

At least one copy of each different letter is printed automatically. In response to the FORMAT prompt "OUTPUT TO PRINTER . . .," enter **3** to specify that you want 3 copies of each letter printed, for a total of 12 letters.

```
                    HAYDEN SOFTWARE COMPANY
                      600 Suffolk Street
                       Lowell, MA  01853

Mr. Timothy Gilbert

Computer Store of Rome
77 Main Street
Rome, N.Y. 99999

Dear Tim,

     Hayden Software Company is pleased to announce the immediate
availabililty of the PIE Writer Word Processing System for the
Apple II, II Plus, Apple //e, and Franklin Ace.

     This powerful addition to the Hayden line of software will
provide your business and professional users with the most
flexible editing and word processing system available.

     No hardware boards or adapters are required for operation of
PIE Writer, yet most of the commercially sold peripherals can be
switched in under software control.  This includes upper and
lowercase adapters, 80-column boards, and 16K memory cards.

     PIE Writer works beautifully with standard Apple II disk
drives and equally well with Corvus and other hard disk systems.

     PIE Writer has the power and flexibility to provide for the
needs of the beginner and the expert.  As your skill and
understanding grow, PIE Writer grows with you.

     All of this is provided at a cost less than that of
comparable word processors, which typically require a Z-80 CPU
card and an 80-column video display board.  This is a real saving
to your customer.  If the user owns a typical 80-column board,
PIE Writer supports it.

     Tim, we would like to answer all of your questions.  Please
call or write for more information.

                              Cordially,


                              Janice Doe
                              Sales Manager
```

Printing Labels from the Same Data File

If you output the primary file above through FORMAT, four copies of the form letter will be printed. You can use the primary file below with the same data file to print a label for each letter's mailing envelope. You must have appropriate labels loaded in your printer, of course, on which to print the labels.

```
.* (File named LABELS)
.*
.* Open file CUSTOMER LIST
.OP CUSTOMER LIST,S6,D2
.*
.* This is a sample of what two output
.* labels will look like:
.*
.* MANAGER'S NAME 1 LABEL START
.* 2
.* STORE NAME 3
.* STREET ADDRESS 4
.* CITY, ST ZIP 5
.* 6
.* MANAGER'S NAME 1 LABEL START
.* 2
.* STORE NAME 3
.* STREET ADDRESS 4
.* CITY, ST ZIP 5
.* 6
.*
.* Set top and bottom margin spacing to zero
.M1 0
.M2 0
.M3 0
.M4 0
.* Set the page length for each label's
.* depth, in this case, 6 lines from
.* label top to label top.
.PL 6
.* Must be in no-fill mode for labels
.NF
.* Read the manager's name
.* and output 1 line space
.RB
```

```
.SP
.* Read the store name and address
.RB
.* Skip to the next label
.NG
.* Line 6 will be skipped when FORMAT
.* starts a new 6-line ''page'' for
.* the next label.
```

Printing the primary file above via FORMAT will produce a label for each of the four form letters. The group terminators and the **.ng** command protect you from printing several inaccurate labels.

PIE Writer's Mail Merge capability, of course, has several applications that have nothing to do with mailing. A standard weekly report that accesses data blocks from any of several data files is just one useful application.

The operator can be queried for the title of the file to be merged simply by inserting an **.op** command in the primary file without naming the data file to be opened. In this way, a source file could be developed that gets a data file's name from the user (via the keyboard) to determine which data file to open.

USER INPUT DURING FORMAT

Simply stated, FORMAT obtains the information it processes in one of two ways. The usual sequence is for FORMAT to process the information in the text buffer. Sometimes this requires FORMAT to find a file stored on diskette or to chain to another file by loading that file into the text buffer. In these cases, once formatting begins, the operator is no longer involved in the process.

The second option allows the operator to provide some information from the keyboard that will be processed either into the output document (the date, perhaps) or to alter the commands (for instance, by selecting single or double spacing).

Typically, there are two specific situations in which it may be desirable to direct FORMAT to display a prompt asking the operator to enter data while FORMAT is processing a file.

Physical Printer Adjustment

Many printers require that a daisy wheel, thimble, or selectric ball be changed to use a different type font for emphasis. In this instance, the escape character and plus sign (]+) are used. This command stops printing immediately and waits for a key to be pressed before continuing. By preceding this command with a **.tm** command, you can create a video prompt string that tells the operator exactly what adjustment must be made to the printer. For instance,

```
.TM CHANGE TO WHEEL 5 AND PRESS ANY KEY
```

or

```
.TM CHANGE TO RED RIBBON AND PRESS ANY KEY
```

will notify the operator to make the necessary change when the]+ is encountered by FORMAT. FORMAT waits, then continues when any key is pressed. If the FORMAT output is directed to a disk file the action of the]+ commnd is deferred until the file is printed. If you are "spooling" your output for later printing, FORMAT replaces the]+ with a CTRL-C in the output. It is the responsibility of the spooling software to recognize this situation and obtain operator actions.

Altering FORMAT Command Variables

The second application for user input during FORMAT allows you both to set formatting instructions and to add lines of text interactively from the keyboard while you are formatting.

Suppose that you wish to create a standard "header" file that allows you to set up a variety of documents for printing. Sometimes you will need a double-spaced draft for easy editing before printing a final single-spaced document. The following command sequence takes these needs into account:

```
.* Set line space command
.* Enter line spacing
.LS ]
.GL Press 1 for final, 2 for draft
```

The] character connects the **.ls** to the **.gl**. It tells FORMAT that the value for the **.ls** command will come from the **.gl**, which will ask the user to input the value at the keyboard. The **.gl** command tells FORMAT to stop processing, provides FORMAT with a video screen prompt, and instructs FORMAT to use the value entered at the keyboard in response to the prompt as the **.ls** variable. According to the prompt above, if the operator enters a **1**, a single-spaced final document will be output, and if a **2** is entered, a double-spaced draft will be produced.

When printing an often-used form letter on a regular basis, you may need to change only a single piece of information, such as the date, each time the form letter is prepared:

```
.* Set up date entry
.TL ''']
.GL Enter today's date (mon dy, yyyy)
```

The above would instruct the operator to place the proper date in the form letter with a simple prompt.

Altering the salutation for each letter could be arranged in the same way:

```
.* Set up salutation entry
Dear ]
.GL Enter name and comma
.SP
Text of letter
```

SHELL FILE STRUCTURES

Shell file structures allow you to perform frequently used Command Processor functions without risking errors due to tedious typing. A shell file contains an exact copy of a sequence of commands that you normally would enter from the keyboard.

To execute a shell file:

1. *Type* **%E filename**
2. *Press* **RETURN**

Every one of the commands in the file will be executed before control is returned to the CP (unless one of the commands requires that the operator enter variables from the keyboard).

This feature makes tasks such as copying and backing up files a simple process. On your master diskette, a shell file called BACKUP has been provided that will do the following:

1. *P-*
2. *CALL $830 (restore working drive)*
3. *"BACKING UP FILE:"?*
4. *%D ?.BAK*
5. *%R ?,?.BAK*
6. *S?*
7. *P+*

The commands are interpreted as follows:

1. *Suppress printing of normal CP prompts*

2. *Restore current data drive, slot, volume*
3. *Print prompt "BACKING UP FILE: FILENAME"*
4. *Delete the file "filename.BAK"*
5. *Rename "filename,filename.BAK"*
6. *Save binary filename*
7. *Enable printing of normal CP prompts*

Shell File Commands

The **P-** command suppresses and the **P+** command enables the normal CP prompts, and they are valid CP commands in themselves.

When a double quote (") is placed in a shell file, all the characters that follow prior to a second double quote are displayed on the video screen.

To print a blank line on the screen, enter two double quotes (" ").

A special feature allows you to change the name of your default file. When an **"I"** is entered at the keyboard, the next line input becomes the default filename. When this is executed within a shell file, the current file can be manipulated and then output with the new filename obtained from the keyboard.

SHIFT KEY MODIFICATION

One of the unfortunate design characters of the original Apple II and Apple II Plus, from a word processing point of view, is that the **SHIFT** key does not give you access to uppercase letters as it does on a standard typewriter. PIE uses the → key as a shift key instead. However, you can make a modification to your Apple that allows the **SHIFT** key to give you access to uppercase when you are in the lowercase mode.

The modification described here is for Revision 7 or more recent Apples Plus computers. It is quite possible to make this modification without soldering. Before following the steps below, however, note that ANY PERMANENT MODIFICATION OF THE APPLE MAY VOID YOUR WARRANTY.

1. *Turn the power off and remove the cover.*
2. *Take a 10-inch piece of light gauge, unstranded wire, strip it at both ends, and bend one end into a hook about a half-inch long.*
3. *Locate the keyboard encoder board to the right underneath the front edge of the cover opening (more or less underneath the keyboard). There are 25 stiff wires connecting the keyboard encoder to the keyboard itself.*

4. *Facing your Apple with the keyboard in front of you and looking at encoder card, locate the second wire from the right, the first being the wire on the extreme right.*

5. *Snake the hooked end of the unstranded wire around this second wire, which is connected to the* **SHIFT** *key. With a piece of (preferably electrical) tape, close the hook so that there is a tight loop around the second wire. The idea is to ensure that the wire does not make contact with any of the other wires on the encoder board.*

6. *Having done this, attach the other end of the wire to pin 4 (pin 1 is in the lower right-hand corner) of the game I/O socket. (If you have one, you could instead attach it to the "lowercase pad" on the ROMPlus card.)*

7. *You now must run PIE CONFIGURE (and FORMAT CONFIGURE if you use the* **.gl** *command) to modify PIE Writer so that it will recognize the* **SHIFT** *key modification you have just made.*

This completes the installation. The **SHIFT** key now will produce uppercase letters when held down with a letter key when in the lowercase mode. In the uppercase mode, the keyboard functions normally, meaning that a **SHIFT-N**, for example, generates the up arrow.

To enter the standard letter-shift characters from lowercase mode, type the following:

To get:	You type:
@	→ * or → :
]	→ > or → .
^	→ ' or → 2

TELECOMMUNICATION

PIE Writer provides a potent telecommunication system.

It is possible to "hookup" with a distant computer via a telephone line and modem. With such a connection established, it is not difficult to obtain the information in another computer's text buffer or to send the contents of your text buffer to another computer. The power of word processing and the immediacy of telecommunication constitute a very desirable information system.

Two computers running PIE Writer can exchange files of information. The process is not complex, but requires one or more pieces of specialized equipment:

1. *A RS-232 serial controller card with an acoustic coupler. Two tested serial cards are the Apple Communications Card and the CCS7710A Serial. Novation's CAT is a tested acoustic coupler.*

or

2. *A Hayes Micromodem II.*

Transmitting Text with a Communications Card

To transmit text via a communications card, enter the Command Processor, and type:

1. **@131<#slot** *(#slot is the slot the Comm Card is in, typically #2)*
2. *Press* **RETURN**

Although nothing appears to happen, this command sets up the "firmware" of the serial card. Either dial the telephone to the distant computer manually and listen for the "carrier whistle," or if your card supports auto-dial, use the appropriate commands to establish the connection.

Now with the terminal connected you may enter control codes that signal commands to the card's firmware. (The card should support a "terminal mode," as the Hayes card does, for example.) At this point, PIE Writer is cut off temporarily from the computer's activities. It remains in memory but no longer participates in the transmission of data.

Use the terminal mode to enter whatever commands the receiving terminal requires to catch the information being transmitted.

Now exit the card's terminal mode and type **CTRL-C** to return the COMMAND?: prompt. (The Ctrl-C corresponds to the 131 entered in the example above. Note in the ASCII reference chart in Appendix A that the decimal value of CTRL-C is 131.)

For example, if you have a Hayes Micromodem II in slot 2, you would do the following:

1. *Type* **@131<#2 RETURN.**. *This connects you with the Micromodem II.*
2. *Type* **CTRL-A CTRL-H**. *This starts the terminal program in half-duplex mode.*
3. *Type* **CTRL-Q** *followed by the telephone number, then* **RETURN** *This instructs the MicroModem to pick up the phone and start dialing.*
4. *When you have established your connection, type* **CTRL-A CTRL-X CTRL-C**. *This exits the terminal mode. The final CTRL-C returns you to the COMMAND?: prompt.*

If you are using the Hayes Micromodem II, note that it only displays its output on the Apple 40-column screen. In general you should use a 40-column version of PIE Writer in conjunction with this card. {However if you have an Apple //e with an Apple //e 80-column card you may use the 80-column version of PIE Writer since it switches back to 40-column mode on input or output from any slot.}

At this point, the Command Processor is active and the Communication Card is inactive. You remain "connected" with the other computer, but for now, your end of the connection is "quiet," while the other end expectantly awaits the transmission in a "stream" of characters.

Load the file you wish to transmit from the diskette into the text buffer, and enter the edit mode by typing:

1. **E $**
2. *Press* **RETURN**

This takes you to the end of the file.

Now we will pick a character, CTRL-C, for instance, and type it in as the last character in the file using the **CTRL-SHIFT-M** {Apple //e and Franklin: use **CTRL-]**} command, followed by the **CTRL-C**. This CTRL-C character acts as a marker. Press **CTRL-SHIFT-P** {Apple //e and Franklin: **CTRL-@**} to return to the CP.

To send the entire contents of the text buffer to the other computer, type the following:

1. **>#slot**
2. *Press* **RETURN**

There will be no indication that information is being transmitted except that the cursor may flash at ragged intervals. The characters will be output to the other computer as if it were a local printer or your video screen. The communications card controls the speed, usually about 30 characters a second.

When the transmission is completed, the "COMMAND?:" prompt reappears. Reenter the terminal code. Do whatever is necessary to verify the text transmitted to the receiving computer.

If during a transmission to a distant computer it takes too long to process a line (more than one "character-time" after it is transmitted), there may be a loss of characters at the other end. PIE Writer cannot send a delay after every carriage return, so it is possible for the receiving computer to lose some of the transmission because it is busy processing its input.

Receiving a File with the Communication Card

It is just as likely that you will want to receive the output of another computer in your PIE text buffer.

In the previous example:

>#slot

sent all the characters out the telecommunications port. To receive characters, you must tell the CP that you want to bring all characters in FROM a slot.

First establish communications with the other computer as in the previous example. Then if you are using a Hayes Micromodem II, and you want to receive lower case characters, you must do the following (assuming the Micromodem is in slot 2):

1. *Type* **CTRL-Y RETURN**. *A "*' will be displayed.*
2. *Type* **6FA:0 RETURN**. *This tells the Micromodem firmware you want to receive lowercase characters.*
3. *Type* **CTRL-Y RETURN.** *This returns you to the COMMAND?: prompt.*

It is necessary for the text material being sent from the distant computer to end with a known marker character that will not appear in the regular text. If the transmitting computer is capable of sending, for instance, a CTRL-C, enter:

1. *Type* **@131<#slot**
2. *Press* **RETURN**

Again, 131 is the decimal ASCII representation of CTRL-C. If there is no card in the slot, the computer will hang, in which case nothing but a **RESET** will return the CP prompt.

This command allows you to receive every charactrer that comes through "#slot" until the character CTRL-C appears, at which point transmission ends. If the CTRL-C is never sent, the Command Processor will wait forever in expectation of that character. If this happens, press **RESET**, and request that the file be retransmitted.

When the transmission is complete, you can edit the transmitted file or save it to diskette. You may disconnect your modem or sign off.

PIE Writer to PIE Writer Communication

The best of all telecommunication environments is when working with two computers equipped with PIE Writer. This allows direct transmission of data from one word processor to another.

It is not possible to see the data being sent or received during transmission. PIE Writer was not designed as a telecommunications device, but its flexibility makes this process possible.

ADDRESS TABLES

Address tables are useful to programmers who want to write utility programs that extend PIE Writer's capabalilties even further. There are several types of addresses that may be of interest:

1. *Page zero pointers, entry points, and special locations.*
2. *The PIE and FORMAT tables normally altered via PIE Configure and FORMAT Configure.*
3. *The keyboard "maps" that tell PIE Writer which function to perform. You may customize the keyboard according to your own needs following the procedure described below under "Command Table Modification."*

All of the above types of addresses follow.

Page Zero Buffer Pointers

$06 Beginning of text ($4000 for 40-column version; $4C00 for 80-column)
$08 End of text + 1 (points to last byte, which contains a 0)
$0A End of text buffer, used only by PIE (default = $95FF)

Important Pie Addresses

$803 Cold start address (can **CALL 2051** from BASIC)
$806 Warm start address (can **CALL 2054** from BASIC)
$809 Restart address (for Apple II's without autostart ROM)
$80C Hook for intercepting commands to CP AUX program
$810 Video slot (0 for 40-column; 3 for 80-column)
$813 FORMAT currently loaded into language card if = 1
$815 High bit set on output to printer if = 1 (default)
$816 Default buffer begin address ($4000 for 40-column; $4C00 for 80-column)
$818 Default buffer end address ($95FF)
$81F Convert break line token (default = CTRL-Z with high bit set)
$820 Printer slot (default = 1)
$821 Line feed after return if = 1 (default = 0)
$822 Printer initialization string (7 characters maximum, plus 0 terminator)
$82A Program disk volume (default = 0)

$82C Program disk drive (default = 1)
$82E Program disk slot (default = 6)
$1290 High bit set in text buffer if = 1 (default)
$1291 Initial bell column (default = 38 for 40-column; 78 for 80-column)
$1292 Initial tab increment (default = 8)
$1293 PPWrap initialization (default = 0 [off]; = PPWrap 2 = Indent)
$1294 Initial lowercase input if = 1 (default)
$1295 Output lowercase to CRT if = 0 (default = 1)
$1296 Address of **SHIFT** key mod input port (default = none)
$1298 Mask for **SHIFT** key bit
$1299 Mask for inverting **SHIFT** key bit
$129A Value OR'ed with control characters to display (= $40 for 40-column)
$129B Absolute value is number of lines to scroll for CTRL-Y and CTRL-N commands; if negative, screen is paged instead of scrolled dynamically
$129C Initial word tab status for manual, PPwrap, and indent modes (default = 0, 1,0)
$12A0-$12DF Command table 1 (commands with no arguments or non-null arguments)
$12E2-$1321 Command table 2 (commands with null arguments and cursor-defined arguments)
$1324-$1333 Character translation table for →key
$3300-$39FF Screen buffer (available to user for executing AUX programs from the Command Processor)

Important FORMAT Addresses

NOTE: In the 64K version of PIE Writer, FORMAT executes out of the language card. In that case, the following variables start at $D003 (instead of $803) in bank 2 of the language card.

$803 Cold start address (can **CALL 2051** from BASIC in the 48K version)
$806 Warm start address (can **CALL 2054** from BASIC in the 48K version)
$809 Restart address (for Apple II's without autostart ROM)
$80F Video slot (0 for 40-column; 3 for 80-column)
$811 Number of lines to scroll before pausing (default = 24)
$813 Printer address (default = $C100)
$816 Output lowercase to CRT if = 1 (default = 0)
$817 Page stop always enabled if = 1 (default = 0)
$818 Default buffer begin address ($4000 for 40-column; $4C00 for 80-column)
$81A Underline mode (0 = none; 1 = BS; 2 = S/S; 3 = Overlay; default = 1)
$81B Initial escape character (default = ']')
$81D Initial data block terminating character (default = '<')
$81E High bit set on output to printer if = 1 (default)

$81F Line feed output after return if = 1 (default = 0)
$820 Initial right adjust if = 1 (default = 0)
$821 Initial fill mode if = 1 (default)
$822 Initial line spacing (default = 1)
$823 Initial margin 1 value (default = 2)
$824 Initial margin 2 value (default = 2)
$825 Initial margin 3 value (default = 1)
$826 Initial margin 4 value (default = 3)
$827 Initial paragraph indent (default = 5)
$828 Initial paragraph indent sign ('+', '-', or ' '; default = '+')
$829 Initial paragraph spacing (default = 1)
$82A Initial paragraph need value (default = 2)
$82B Initial page length (default = 66)
$82C Initial line length (default = 65)
$82D Initial page offset (default = 0)
$82E Initial offset character (default = ' ')
$830 Underline start sequence (3 characters maximum, plus 0 terminator)
$834 Underline stop sequence (3 characters maximum, plus 0 trerminator)
$838 Boldface start sequence (3 characters maximum, plus 0 terminator)
$83C Boldface stop sequence (3 character maximum, plus 0 terminator)
$840 Printer initialization string (7 characters maximum, plus 0 terminator)
$848 Printer type (1 = Diablo; 2 = Qume; 3 = NEC; 4 = Epson; 5 = Centronics; 6 = other; default = 6)
$849 Width of character 1 120's for daisy-wheel printers (= 12 for 10 characters-per-inch)
$850 Initial lowercase input (**.gl**) if = 1 (default)
$851 Boldface mode (0 = none; 1 = BS; 2 = S/S; default = 1)
$852 Number of strikes per character for BS boldface (default = 3)
$853 Stop character on output to disk (default = CTRL-C)
$854 Line feed after carriage return on output to disk if = 1 (default = 0)
$855 Address of **SHIFT** key mod input port (default = none)
$857 Mask for **SHIFT** key bit
$858 Mask for inverting **SHIFT** key bit
$859 Value OR'ed with control characters to display (= $40 for 40-column)
$85A Program disk volume (default = 0)
$85C Program disk drive (default = 1)
$85E Program disk slot (default = 6)

Addresses Common to PIE and FORMAT

$3F40 Remembered filename, terminated by $8D
$3F76 Remembered buffer end address

$3F80 Remembered tab settings
$4000 Beginning of text buffer (40-column)
$4C00 Beginning of text buffer (80-column)

Locations $F8-$FF and $300-$3CF are reserved for the user (such as for a printer driver). All other memory below the top of the text buffer (for instance, $95FF) is reserved for use by PIE and FORMAT, except for the AUX program buffer at $3300 as described above.

FORMAT Limits

- *Maximum page length: 250 lines*
- *Maximum page number: 9999*
- *Maximum output line length: 132 characters*

Command Table Modification

The keyboard layout for PIE is all table-driven, and it is possible to reconfigure the keyboard. There are two command tables indicated above: one for commands with no arguments or with a non-null argument, and the second for commands with a null argument. The addresses for each are listed in the PIE address tables.

Each table has 32 addresses, one for each control character (0-$1F). To rearrange the keyboard, just swap addresses corresponding to the appropriate keys. If the addresses in both tables for the same key are the same, then both tables should be modified together.

Custom Program Area

The area of the computer's memory that is dedicated to the screen display in edit mode is free when you are in the PIE Command Processor. When a "%" command is encountered, the PIE AUX file is loaded into this area ($3300-$39FF).

If you want to write a utility or set of utilities using the address tables above, go right ahead. All code should reenter PIE Writer through the DOS $3D0 softstart vector. Use the **BLOAD** and **CALL** or **BRUN** commands to execute your utility.

Custom Printer Driver

The beginning of page 3 of memory ($300-$3CF) has been reserved for a custom printer driver. After you have written your driver, run FORMAT Configure (selection 4 in the System Menu) to inform FORMAT of the address of the driver.

The driver will get control at the address specified in FORMAT Configure with

the first character to be printed in the accumulator (A register). It should perform any initialization needed, and then print the first character. If you need to specify a different address to be entered for the remaining characters to be printed, the driver must modify CSW (locations $36 and $37) before returning. The driver should return with an RTS instruction. It does not need to save any registers. In addition to page 3 memory, page 0 locations $F8-$FF are available for use as variables.

You may want to modify the file PIE WRITER: WORD PROCESSING to add a statement to BLOAD the driver every time the PIE Writer disk is booted.

Appendix A

ASCII Character Codes

The decimal (Dec) and hexadecimal (Hex) codes in the tables below are for standard ASCII character codes with the high-order bit off. By convention, both the Apple and Franklin usually store characters with the high bit set instead. PIE Writer's default mode is to likewise store characters with the high bit set in the buffer (see Chapter 4, PIE Configure, if you have need to change this).

To convert from standard ASCII to high-bit set ASCII, add either 128 to the decimal value, or $80 to the hex value. For example, the standard CTRL-D (EOT) character is decimal 4 (and also hex 4). But on the Apple and Franklin, CTRL-D is represented as 132 (decimal) or $84 (hex).

CONTROL CHARACTERS

Character		Dec	Hex	Translation (for CTRL-SHIFT-M command)
CTRL-@	(NUL)	0	00	
CTRL-A	(SOH)	1	01	
CTRL-B	(STX)	2	02	
CTRL-C	(ETX)	3	03	
CTRL-D	(EOT)	4	04	
CTRL-E	(ENQ)	5	05	
CTRL-F	(ACK)	6	06	
CTRL-G	(BEL)	7	07	
CTRL-H	(BS)	8	08	
CTRL-I	(HT)	9	09	
CTRL-J	(LF)	10	0A	
CTRL-K	(VT)	11	0B	
CTRL-L	(FF)	12	0C	
CTRL-M	(CR)	13	0D	
CTRL-N	(SO)	14	0E	
CTRL-O	(SI)	15	0F	
CTRL-P	(DLE)	16	10	
CTRL-Q	(DC1)	17	11	
CTRL-R	(DC2)	18	12	
CTRL-S	(DC3)	19	13	
CTRL-T	(DC4)	20	14	
CTRL-U	(NAK)	21	15	
CTRL-V	(SYN)	22	16	
CTRL-W	(ETB)	23	17	
CTRL-X	(CAN)	24	18	
CTRL-Y	(EM)	25	19	
CTRL-Z	(SUB)	26	1A	
CTRL-[	(ESC)	27	1B	;
CTRL-\	(FS)	28	1C	<
CTRL-]	(GS)	29	1D	= or CTRL-SHIFT-M
CTRL-△	(RS)	30	1E	> or CTRL-SHIFT-N
CTRL-_	(S)	31	1F	?

PRINTING CHARACTERS

Character	Dec	Hex	Translation (use → key)
SP	32	20	
!	33	21	
"	34	22	
#	35	23	
$	36	24	
%	37	25	
&	38	26	
'	39	27	
(	40	28	
)	41	29	
*	42	2A	
+	43	2B	
,	44	2C	
-	45	2D	
.	46	2E	
/	47	2F	
0	48	30	
1	49	31	
2	50	32	
3	51	33	
4	52	34	
5	53	35	
6	54	36	
7	55	37	
8	56	38	
9	57	39	
:	58	3A	
;	59	3B	
<	6	3C	
=	61	3D	
>	62	3E	
?	63	3F	
@	64	40	

Character	Dec	Hex	Translation (for CTRL-SHIFT-M command)
A	65	41	
B	66	42	
C	67	43	
D	68	44	
E	69	45	
F	70	46	
G	71	47	
H	72	48	
I	73	49	
J	74	4A	
K	75	4B	
L	76	4C	
M	77	4D	
N	78	4E	
O	79	4F	
P	80	50	
Q	81	51	
R	82	52	
S	83	53	
T	84	54	
U	85	55	
V	86	56	
W	87	57	
X	88	58	
Y	89	59	
Z	90	5A	
[	91	5B	<
\	92	5C	/
]	93	5D	>
△	94	5E	"
–	95	5F	=
‘	96	60	'
a	97	61	
b	98	62	
c	99	63	
d	100	64	
e	101	65	
f	102	66	
g	103	67	
h	104	68	

Character	Dec	Hex	Translation (for CTRL-SHIFT-M command)
i	105	69	
j	106	6A	
k	107	6B	
l	108	6C	
m	109	6D	
n	110	6E	
o	111	6F	
p	112	70	
q	113	71	
r	114	72	
s	115	73	
t	116	74	
u	117	75	
v	118	76	
w	119	77	
x	120	78	
y	121	79	
z	122	7A	
{	13	7B	(
|	124	7C	!
}	125	7D	)
~	16	7E	+
DEL	127	7F	#

Appendix B

PIE WRITER: Changes from Previous Versions

CHANGES FROM VERSION 2.1

This release of PIE Writer (Version 2.2), while functionally identical to Version 2.1, significantly expands the hardware PIE Writer can make use of, while making its configuration to any particular hardware automatic for most users.

Version 2.2 now supports: both the Apple II and II Plus with either 48K RAM or 64K RAM ("language" or "RAM" card); the Apple //e (64K); and the Franklin Ace 100 and 1000 (both 64K).

Version 2.2 also supports many more 80-column cards, including the Apple //e 80-column card.

PIE Writer Version 2.2 offers significant performance enhancements when configured for 64K RAM: The entire word processing system—the PIE Editor, FORMAT, and the "AUX" files—is loaded into memory at once, meaning the program disk is no longer needed after the initial load except to load the Help file and to run the configure programs. This means PIE Writer can be effectively used in a single-drive 64K system: after booting up and selecting PIE (choice 1 in the System

Menu), and optionally loading the Help file using the %H command, you can remove the PIE Writer program disk—you won't need to access it again unless you return to the System Menu.

Each copy of PIE Writer now comes with two disks: one containing 40/80 column and 48K/64K versions of PIE Writer for the Apple II and II Plus, and the second disk containing both 40 and 80-column versions for the 64K Apple //e and Franklin.

Finally, the documentation for this version has been newly typeset, correcting errors in the previous version, and adding additional information, including an expanded index.

CHANGES FROM VERSION 2.0

Apple PIE Version 2.0 users will be able to use PIE Writer right away: PIE Writer keeps virtually all of the Version 2.0 features intact, while adding significant enhancements. To get the most out of your new copy of PIE Writer, take note of the following changes. For further information on new features, see the appropriate sections of this manual.

PIE Command Processor

All files created by Apple PIE 2.0 can be read and edited by PIE Writer, after converting them to 16 sector files using the Apple-supplied MUFFIN program.

The most obvious change to the Command Processor is that the E command is now used to edit an existing file, while an N command is used to create a new file. The new E command optionally allows a starting line number (with $ used to mean the last line number in the file). The old E and R commands were changed to avoid ambiguity between edit and re-edit.

The Command Processor now displays the remembered filename, length and memory left after saving and loading files, and when re-entering the CP. This information is also available using the modified LE command. It's the primary filename that is remembered: saving and loading "side files" using >, << or line number range (begin,end) does not affect the remembered filename.

The CATALOG command has been shortened to C to save you keystrokes. Slot, drive and volume may still be optionally specified.

New CP commands include CALL machine language routine (which replaces the %U user function) and a CTRL-Y exit and return to and from the Apple monitor. Several new auxiliary commands (%) were added: %C outputs not only a catalog but

also the number of sectors left on a disk; %FS displays a complete file status, including word and line counts; %TS and %TL save and load the PIE tab settings; %B executes a BACKUP shell to backup the current file; %H enables the help screen (it loads it into memory); and %LF toggles the line feed after return flag (for switching between peripheral devices set up in two different ways).

The quit (Q) command is now %Q, and a %M command was added to return to the main menu. The CI and CN commands are now invoked by %CI and %CN respectively. Three new auxiliary commands (%ST, %SP, and %RS) were added to interface with the Double Time Printer package available for the Apple II and II Plus. The cassette load and save commands (%LO and %SA), however, have been removed.

To minimize the impact on Version 2.0 shell files, the ! command (which is no longer documented) can still be used to display the current filename, and is identical to the modified ? command in the current version. The file management commands (%D, %L, %U, %V, %R, and %E) no longer require (but still allow) a CTRL-D after the %.

Whenever the program disk is accessed (when, for example, FORMAT is loaded or any CP auxiliary command is issued or you return to PIE from FORMAT) the default drive is automatically restored afterwards.

An improved printer interface is now available from the CP: options include a printer initialization string, line feed after carriage return option, and high bit set or rest.

PIE Editor

The key assignments all remain the same, with the single exception that CTRL-SHIFT-N now removes one word instead of one line (the latter can still be performed using ESC CTRL-SHIFT-N). On the system status line, the names PPMODE and AIMODE were changed to PPWRAP and INDENT, respectively. INDENT mode is now entered using ESC I RETURN, but ESC A RETURN has also been retained for compatibility with Version 2.0. ESC H RETURN is used to display the help screen.

Word tabbing and word delete are two major additions to the PIE editor: word tabbing (using CTRL-A and CTRL-G) is enabled by default in PPWRAP, but may be toggled between word tabbing and column tabbing in any of the three editor modes using ESC W RETURN. (Each editor mode remembers the type of tabbing you set for it). Tab settings are cleared from the top border when word tabbing is in effect. Word delete (CTRL-SHIFT-N) is available in any mode. Deletion up to a specified character is avialable using ESC c CTRL-J.

The only change to an existing command is that typing CTRL-S in column 1 now wraps to the last character on the previous line, instead of the last column.

Two undocumented Version 2.0 features are now fully documented in this manual: using wildcards in search strings (CTRL-Z and CTRL-Q), and entering any control character into a file using CTRL-SHIFT-M.

Moving or removing lines using line number (ESC #mm,nn CTRL-O or CTRL-SHIFT-N) now requires the program disk in the 48K version of PIE Writer: the command must be loaded from the PIE AUX file.

The PIE editor now has a 31-character type-ahead buffer, allowing text or commands to be typed even when the system status line displays BUSY.

When the editor is ALMOST OUT OF MEMORY, the bell is rung every time a charcter is typed (except on the ARG line) to minimize the possibility of creating files too large for the text buffer, and worse, ultimately overwriting DOS.

After exiting PIE and then re-entering again (using the E command in the CP), PIE will return to the last line number and column except when a line number is specified after the E command. This is particularly useful in conjunction with the backup (%B) command: you can go to the CP, make a backup to disk, and with just two keystrokes, return to where you left off.

FORMAT TEXT Processor

Any files created by Apple PIE 2.0 can be read and formatted by FORMAT, after converting them to 16 sector files. All Version 2.0 FORMAT commands are supported by the new version, so existing files should format the same as before.

The "remembered filename" and tab settings are now preserved when switching from PIE to FORMAT and back. FORMAT can now input either binary or text files, as either the main text file or a data file. There is no longer any limit to the size of the data file. Input from cassette tape is no longer supported.

The formatted text can be output to a text file on disk, in addition to being output to the printer or screen. When text is formatted to a disk file, line feed after return is output as an option. The output representation of the new stop character (]+) can also be specified.

Paging to the screen is now done 24 screen lines at a time, instead of being based on the page boundaries in the document. Page boundaries are now displayed as a series of dashes when output to the screen. Page numbers can go as high as 9999.

The page offset is no longer output when formatting to the screen. When text is output to the printer or disk, the "initial offset character" is only output if it is not a space.

Several new commands were added: .tl generates a title anywhere on the page, .bf bold-faces the next input line, .cp capitalizes the new input line, .tm displays a message

on the screen, .ng is used to synchronize form letter data file blocks, and .* allows insertion of comments.

The .cn, .nc, and .sc commands are included for compatability with Version 2.0 files but are not documented in this manual. The .gl command now gets lower case input, using the → key to switch to upper case (or if installed, the shift key mod is used).

Two new escapes have been added:]+ stops on a character and waits, and]" sends a "zero-width" control string to the printer which is ignored by the justificiation routines. Except for the backspace character, which has a width of -1, all control characters also have zero width.

Underlining can now be performed three ways: using backspaces, start/stop characters, or overlay. The latter two modes were added to support the Centronics 737/739 and Epson MX-80 printers. Boldface can be done in two ways (either overstriking using backspaces or using start/stop characters.

A printer initialization string (up to 7 characters) can now be sent to the printer at the beginning of a document. The default printer slot has been changed to slot 1. And PIE Writer offers incremental spacing for "daisy-wheel" printers, such as the Diablo, Qume, and NEC.

On the RETURN TO PIE/RE-RERUN FORMAT prompts, %Q and %M and CTRL-Y function the same as in the PIE CP.

Other Changes

The CTRL-J was removed from the end of the "hello" program PIE WRITER: WORD PROCESSING and two null filenames added to the disk catalog to separate the PIE Writer files.

The SYSGEN PIE and SYSGEN FORMAT programs have been renamed PIE CONFIGURE AND FORMAT CONFIGURE. Both require APPLESOFT BASIC in ROM or on the language card. But PIE and FORMAT can still be run without Applesoft.

PIE Writer now requires a minimum of 48K RAM. Both Pie and FORMAT now start at $803 (2051) to avoid being clobbered by Applesoft. A warm start at $806 (2054) was also added. The RESET key vector for the autostart ROM is now initialized. For users without the autostart ROM, the old PIE reset vector at $803 is now located at $809.

The beginning and ending buffer pointers were moved to locations $6-$9 to stay clear of both Applesoft and Integer BASIC. Existing default variables in the $800 page were all moved up by 16 bytes ($10). However, the text and screen buffer locations were not changed: the text buffer in the 40-column version is still located at $4000-$95FF,

and in the 80-column versions at $4C00-$95FF. The screen buffer starts at $3300 in both versions. The PIE command tables are also still in the same location, but of course the contents have changed.

The lower part of page 3 of memory is no longer used by PIE Writer, so locations $300-$3CF are available for a user printer driver or other utilities. Page 0 locations $F8-$FF have also been reserved for user variables when calling these routines from PIE or FORMAT.

Index

Adjusted text, 113
Appending files, 100
Apple II and II Plus, 7, 9, 12, 15, 19, 64, 91, 141
Apple //e, 7, 10, 15, 19, 79e, 144
Argument, 79
ARG, 79
ASCII end of file marker, 101
ASCII reference table, 152
Automatic repeat, 17
Auxiliary commands (command processor), 102
  %B, 105
  %C, 104
  %CB, 64, 106
  %CI, 106
  %CJ, 106
  %CN, 107
  %CS, 106
  %D, 107
  %E, 107
  %FS, 104
  %H, 14, 91, 104
  %L, 107
  %LC, 105
  %LF, 63, 107
  %M, 36, 105, 109
  %Q, 36, 104
  %R, 107
  %RS, 108
  %SP, 108
  %ST, 108
  %TL, 106
  %TS, 105
  %U, 107
  %V, 107
  %WC, 105
AUX Files, 102
Backing up your PIE Writer diskette, 7
Backspacing, 71, 72
Backup, 105, 140
BASIC,
  Breaking long lines, 106
Begin page, 48, 114
Beginning of file, go to, 83
Beginning of line, 18, 80
Bell column, 65, 81
Binary files, 100
Blank lines, 44, 48, 114
Block move, 92
Block terminating character, 73, 120, 128, 134
Body of letter, 45
Bolface, 58, 72, 118
Booting the disk, 7, 128
Bottom home, 16, 80, 93

Bottom margin, 55, 56, 57, 75, 117
Break in filling, 110, 113
C(atalog) command, 28, 98
CALL command, 99
Catalog and space on disk, 104
Centering, 58, 118
Centronics, 70
Chaining files, 119
Character translation, 95, 152
Closing, 46
COLUMNS file, 32
Command Tables, 149
Commands, 5, 79
Comment lines, 125
Compound commands, 79
Conditional page, 48, 114
Conditional replace, 89
Configure, 58, 61, 68
Control characters, entry, 95
  as commands, see editor commands
Convert break token, 64
Copying lines, 86, 93
CP AUX File, 102
Ctrl-c,
  as stop character, 73
Ctrl-x,
  cancels CP input line, 26
Ctrl-y,
  from CP, 99
Ctrl-z,
  convert break token, 64
Current filename, 27
Cursor, 3, 15
Cursor movement, 16, 80
  destructive, 20, 85
  in CP using ESC, 97
Cursor-defined commands, 92
Daisywheel printers, 70
Data files, 120, 131
Dates, 44, 139
Defaults, 5
  FORMAT, 35, 109
Deleting,
  a character, 84
  a file, 107
  a line, 84, 93
  a word, 84
DELETE key, 20e, 84e
Destructive cursor moves, 20
Diablo, 70
Disk space, 104
DOS
  Commands, 107
  Errors, 128
Dot commands,
  .ad, 46, 113
  .bc, 127
  .bf, 58, 118
  .bl, 48, 114
  .bp, 48, 114
  .br, 45, 113
  .ce, 58, 118
  .cp, 58, 119
  .ec, 126
  .fi, 34, 39, 44, 113
  .fo, 53, 115
  .gl, 73, 123, 139
  .he, 53, 115
  .in, 44, 50, 115
  .li, 127
  .ll, 35, 115
  .ls, 47, 114
  .m1, 55, 75, 117
  .m2, 56, 75, 117
  .m3, 56, 75, 117
  .m4, 56, 75, 118
  .na, 46, 113
  .ne, 39, 48, 114
  .nf, 34, 44, 113
  .ng, 122
  .np, 39, 46, 111
  .nx, 119

.op, 120
.pi, 40, 51, 111
.pl, 48, 114
.pn, 40, 111
.po, 42, 114
, 39, 51, 111
.ps, 39, 111
.rb, 120
.sb, 121
.sn, 125
.sp, 39, 44, 114
.st, 124
.ti, 39, 51, 115
.tl, 53, 116
.tm, 124
.ul, 58, 118
.*, 125
Doubletime Printer, 67, 108
E(dit) command, 27, 98
Editor commands
(standard keyboard),
ctrl-a, 17, 25, 81
ctrl-b, 18, 80
ctrl-c, 16, 80
ctrl-d, 16, 80
ctrl-e, 16, 80
ctrl-f, 16, 80
ctrl-g, 17, 25, 81
ctrl-i, 84, 86
ctrl-j, 20, 84
ctrl-k, 86, 87
ctrl-l, 87, 92
ctrl-n, 19, 83
ctrl-o, 87
ctrl-p, 21, 84
ctrl-q, 88
ctrl-r, 19, 22, 82
ctrl-s, 16, 80
ctrl-shift-m, 96
ctrl-shift-n, 84
ctrl-shift-p, 22, 25, 32, 97
ctrl-t, 15, 22, 83
ctrl-v, 15, 18, 22, 82
ctrl-w, 89
ctrl-x, 89
ctrl-y, 19, 22, 83
ctrl-z, 88
return, 81, 90
→, 19, 96
←, 20, 84
Editor commands
(Apple //e keyboard),
ctrl-a, 18e, 25e, 81e
ctrl-b, 18e, 80e
ctrl-c, 88e
ctrl-d, 20e, 84e
ctrl-e, 84e, 86e, 92e
ctrl-g, 16e, 80e
ctrl-h, 16e, 80e
ctrl-i, 18e, 25e, 81e
ctrl-j, 16e, 80e
ctrl-k, 16e, 80e
ctrl-l, 87e, 92e
ctrl-n, 19e, 83e
ctrl-o, 87e
ctrl-p, 21e, 84e
ctrl-q, 88e
ctrl-r, 19e, 22e, 82e
ctrl-s, 20e, 84e
ctrl-t, 15e, 22e, 83e
ctrl-u, 16e, 80e
ctrl-v, 15e, 19e, 22e, 82e
ctrl-w, 89e
ctrl-x, 89e
ctrl-y, 19e, 22e, 83e
ctrl-z, 88e
ctrl-@, 22e, 96e
ctrl-\, 86e, 92e
ctrl-], 96e
ctrl-^, 85e, 92e
delete key, 20e, 84e
return, 81e, 90e
tab key, 18e, 25e, 81e
↑, 16e, 80e

↓, 16e, 80e
→, 16e, 80e
←, 16e, 80e
Editing, 27
End of file marker, 101
End of file, go to, 83
End of line, 18, 80
Entering control characters, 95
Entering non-keyboard characters, 95
Epson, 70
Errors, 128
  memory, 26, 65
  messages, 127
  starting, 8
ESC h, 15, 91, 104
ESC key with compound commands, 79
Escape character in FORMAT, 73, 126
Exec'ing shell files, 107, 141
Exiting the text editor, 22, 97
F(ORMAT) command, 33, 98
File chaining, 119
File lengths, 26, 99, 104
File status, 26, 104
File switching, 119
Filenames, 25
  remembered, 27
Fill mode, 34, 76, 111, 113
Finding text, 88
Foot title, 53, 116
Form letters, 119, 131
FORMAT Configure, 68
Formatting from diskette, 33, 108
Franklin ACE, 7, 11, 15, 19
Global search and replace, 89
Go to
  a line, 83
  beginning of file, 83
  end of file, 83
Graftrax, 70
Hanging indent, 51, 52, 126
Hard disk, 11, 67
Hardware, 6
  see also
    Apple II and II Plus,
    Apple //e,
    Franklin ACE,
    lower case,
    shift key modification,
    80-column boards
Head title, 53, 116
Headings, 44, 53, 56, 116
Help screen, 15, 91, 104
High order bit
  in buffer, 63
  on output, 64, 71
Home and bottom home, 16, 80, 93
I in shell command, 141
Indent mode, 90
Indentation, 41, 44, 50, 115
Initial bell column, 65
Initial fill column, 76
Initial offset character, 74
Initial right adjust, 76
Initial tab increment, 66
Input from slot, 101
Input line joining, 127
Insert mode, 85
Inserting, 84
  a blank line, 84
  a character, 21
Inside address, 45
Joining lines
  in CP, 106
  in FORMAT, 127
  in PIE, 86
Justification, 46, 76, 113
  with 0 width character string, 127
Keyboard input to FORMAT, 123, 138
L(oad) command, 100

Labels, 137
LE(ngth) command, 99
Left hand margin, 42, 114
Lessons
  FORMAT, 31
  PIE, 15
Line count, 105
Line feed, 63, 71, 107
Line feed output to disk, 73
Line, go to a, 83
Line length, 43, 74, 115
Line spacing, 47, 74, 113
List indent, 51, 52, 126
Literal lines, 127
Loading files, 14, 28, 100
  from FORMAT, 33, 109
Long lines, 106
Lower case
  adapter, 9, 64, 73
  display, 19, 63, 64, 91, 72,
  entry, 20, 72, 96
M(emory) FORMAT option, 109
Mail merge, 119, 131
Manual mode, 24, 66, 90
Margin, editor, 4, 17
Margins, 42
  left and right, 42, 43, 114
  top and bottom, 55, 56, 57, 75, 117
Maximum output line, 149
Maximum page length, 149
Maximum page number, 149
Memory error, 26
Modem, 143
Modifers, 79
Moving lines, 86
N(new) command, 23, 98
NEC Spinwriter, 70
Need value, 48, 76, 114
New files, 23, 98
New page, 48, 114
No adjust, 113
No-fill mode, 35, 111, 113
No-fill paragraphs, 39, 113
Outdents, 51, 52, 126
Outline, 50
Output
  FORMAT output to disk, 108
  to disk, 37, 73
  to printer, 37, 108
  to slot, 101
P+, 141
P-, 141
P(rinter) FORMAT option, 37, 108
Page length, 48, 74, 114
Page limits, 108
Page number token, 117
Page numbers, 55, 117
Page offset, 74, 114
Page stop enabled, 70
Paper size, 41
Paragraph indent, 41
Paragraphs, 39, 76, 110
PIE AUX File, 149
PIE commands, see editor commands
PIE Configure, 61
PIE help, 14
Plus and minus 'n' values, 110
PPWrap mode, 24, 46, 66, 81
Primary file, 132
Printer adjustment, 138
Printer backspace, 70
Printer command characters, 71, 72
Printer driver
  in FORMAT, 69, 149
  IN CP, 102
Printer initialization, 63, 69
Printer interface
  PIE, 62
  FORMAT, 69
Printer special features, 11, 58, 127

Printer slot
  PIE, 62
  FORMAT, 69
Printer type, 70
Printing
  labels, 137
  using FORMAT, 36
  using PIE, 37
Program disk drive, 66
Quiting from the CP, 103
Qume Sprint, 70
Reading a range, 101
Reading text files, 5, 23
Recalling lines, 87
Rename file, 107
Replacing text, 89
REPT key, 16
RESET key, 128
Right adjust, 113
Right hand margin, 42, 114
Right side of the screen, 4, 17
S(ave) command, 100
Salutation, 45
Saving
  files, 25, 47, 100
  FORMAT Configure changes, 77
  PIE Configure changes, 68
  tab settings, 105
Scrolling, 4, 1882
Search and replace, 88
Shell files, 140
Shift key, 19, 29, 65, 92, 141
Shift key modification, 9, 19, 64, 73, 141
Shifting text, 21
  with cursor-defined commands, 92
Signature, 46
Simple commands, 79
Single-sheet paper, 72
Slot/drive specification, 37, 67
SPACE bar, 20
Space on diskette, 104
Special characters, 63
  entry, 95
Splitting lines
  in CP, 106
  in PIE, 86
Spooling with Doubletime, 108
Start/Stop sequences, 71
Startup, 6
Status line, 17
Synchronizing data files, 122
System Configure, 8
System disk, 7, 67
System menu, 14, 105
System monitor, 98
Tabs, 17, 66,
  loading and saving, 105, 106
TAB key, 18e, 25e, 81e
Telecommunication, 142
Temp*** file, 120
Temporary indent, 41, 53, 115
Terminal communication, 138
Text buffer, 23
  contents destroyed, 23
  end, 65
  recovery, 128
Text entry modes, 24, 82, 90
Text files, 100
Titles, 53, 116
Toggles, 16, 18, 21, 80
Top margin, 55, 56, 57, 75, 117
Translation characters, 142, 152
Transmitting text, 143
Tutorial
  FORMAT, 31
  PIE, 15
Type-ahead, 84
Typing modes, 24, 82, 90
Underlining, 58, 71, 118
Unpaddable space, 126
Upper register characters, 19

Uppercase
  display, 19, 91
  entry, 19, 92
Version 2.1, 157
Version 2.2, 157
Viewing files in FORMAT, 47, 109
Wild card search, 88
Window, 3
Word count, 105
Word delete, 84
Word tabbing, 25, 82
Writing a range, 101
Writing text files, 101
Zero-width characters, 127
80-column boards, 3, 10, 12

$ special token, 86, 88
%, Page number token, 117
→, see editor commands
← , see editor commands
<,
  block terminating char, 73
  in CP, 101
>, 37, 100
?, 99
? command, 28, 99
] as escape character, 73, 126
]>, 126
]+, 73, 127
]", 127
]return, 127

This type for his book was set using PIE Writer to send text to Hayden's in-house AM Varityper Comp/Set typesetting system. Finished books were off press two and a half weeks after the transmission began. The majority of typesetting commands, including font changes, were imbedded using PIE Writer prior to transmission. A small machine-language subroutine, invoked from PIE Writer's command processor with a CALL statement, managed the interface between the Apple and the typesetting system.

Contents

1
Introduction to PIE Writer

HOW TO USE THIS MANUAL 2
BASIC CONCEPTS 2
- The Cursor 3
- The Window 3
- Beyond the Right-Hand Margin 4
- Scrolling 4
- Commands 5
- Default Values 5
- Final Suggestions 5

GETTING STARTED 6
- Backing Up Your PIE Writer Diskettes 7
- Booting the PIE Writer Diskette 7
- If Additional Help Is Needed 8
- System Configure 8
- Notes About The System Configure Program 11

2
The PIE Text Editor and Command Processor

HOW TO USE THE TEXT EDITOR 13
PIE TEXT EDITOR LESSONS FOR THE STANDARD KEYBOARD 15
- Lesson 1: The Cursor 15
- Lesson 2: Home and Bottom Home 16
- Lesson 3: Simple Cursor Movement 16
- Lesson 4: Cursor Movement Practice 16

Lesson 5: The Status Line 17
Lesson 6: Beyond the Right-Hand Margin 17
Lesson 7: Tabs 17
Lesson 8: Moving the Cursor to the Beginning or End of a Line 17
Lesson 9: Scrolling Pages and Lines 18
Lesson 10: Upper and Lowercase Character Display 19
Lesson 11: Uppercase Text Entry 19
Lesson 12: Entering Upper and Lowercase Characters 20
Lesson 13: Destructive Cursor Moves 20
Lesson 14: Deleting a Character 20
Lesson 15: Inserting a Character 21
Lesson 16: Shifting Text Left or Right 21
Lesson 17: Returning to the Beginning of a File; Leaving the Text Editor and Entering the Command Processor 22
THE COMMAND PROCESSOR 22
Saving the Text Buffer 23
Beginning a New File 23
Typing Modes 24
Saving a File 24
A Caution About File Lengths 25
Editing a File 26
Displaying Your Filename 27
Viewing the Catalog of Files 27
Loading a File 28
PIE Summary 29

3
The FORMAT Text Processor

LESSON 1: INTRODUCTION 31
Entering FORMAT Dot Commands 31
Displaying a File 33
Reading Formatted Text on the Screen 34
Other FORMAT Commands 35
LESSON 2: PRINTING A FILE 35
Transferring Between the Text Processor, the Text Editor, and the System Menu 35
Printing Hard Copy 36
LESSON 3: PARAGRAPH CONTROL 39
Standard Paragraphs 39
Other Paragraph Commands 39
One-Time Paragraph Indent 41
LESSON 4: DESIGNING A LETTER 41
Calculating Margin Widths 42
Setting Up the Left Margin 42

Line Length 43
Right Margin 43
The Heading and Date 44
Inside Address and Salutation 45
Body of Letter 45
Closing and Signature 46
Saving and Reviewing the Letter 47
LESSON 5: INDENTATION 48
Extract Indent 48
List and Outline Indent 48
LESSON 6: RUNNING TITLES, PAGE NUMBERS, MARGINS, AND FINISHING TOUCHES 53
Running Titles 53
Page Numbers 55
Margins 55
Finishing Touches 58

4
PIE and FORMAT Configure

PIE CONFIGURE 61
Printer Interface 62
Lowercase and Special Characters 63
Other PIE Editor Defaults 65
Program Disk Slot, Drive, Volume 66
Doubletime Spooler Option 67
Saving PIE Configure Options 68
FORMAT CONFIGURE 68
Printer Interface 69
Lowercase and Special Characters 70
Page Format 74
Top and Bottom Margins 75
Paragraphs and Filling 76
Program Disk Slot, Drive, Volume 77
Saving FORMAT Configure Options 77

5
Reference Section

PIE TEXT EDITOR 79
Simple Cursor Movement 80
Cursor Jumps 80
Tabbing 81
Line and Window Scrolling 82
Go To a Line 83

Inserting and Deleting 84
Splitting and Joining Lines 85
Copying and Moving Lines 86
Search and Replace 88
Text Entry Modes 90
Displaying the Help Screen 91
Displaying and Entering Upper-and Lowercase Letters 91
Cursor-Defined Commands 92
Entering Control and Nonkeyboard Characters 95
Exiting the Editor and Transferring to the Command Processor 96
PIE COMMAND PROCESSOR 97
Control Commands 97
Input/Output Commands 99
Auxiliary Commands 102
FORMAT TEXT PROCESSOR 108
FORMAT DOT COMMANDS 109
Paragraphs 110
Layout 112
Indentation 115
Running Titles and Margins 116
Centering, Underlining and Boldface 118
File Switching and Form Letters 119
Interactive Input and Control 123
Character Translation and Definition 125
ERRORS AND ERROR MESSAGES 128
Reset Error 128
Recovering Text After Rebooting 128
Disk Errors 128

6
Advanced Topics

MAIL MERGE 131
Primary File 132
Data File 134
Formatted Output 135
Printing Labels from the Same Data File 137
USER INPUT DURING FORMAT 138
Physical Printer Adjustment 138
Altering FORMAT Command Variables 139
SHELL FILE STRUCTURES 140
Shell File Commands 141
SHIFT KEY MODIFICATION 141

TELECOMMUNICATION 142
Transmitting Text with a Communications Card 143
Receiving a File with the Communication Card 145
PIE Writer to PIE Writer Communication 145
ADDRESS TABLES 146
Page Zero Buffer Pointers 146
Important PIE Addresses 146
Important FORMAT Addresses 147
Addresses Common to PIE and FORMAT 148
FORMAT Limits 149
Command Table Modification 149
Custom Program Area 149
Custom Printer Driver 149

Appendix A
ASCII Character Codes

CONTROL CHARACTERS 152
PRINTING CHARACTERS 153

Appendix B
PIE WRITER: Changes from Previous Versions

CHANGES FROM VERSION 2.1 157
CHANGES FROM VERSION 2.0 158
PIE Command Processor 158
PIE Editor 159
FORMAT Text Processor 160
Other Changes 161

INDEX 163

If you are working in the Text Editor with the lesson, practice the commands as they are introduced. Enter text anywhere on the screen. If you go beyond the right-hand margin and the window shifts to the right, press **RETURN** to get back to the main page.

To begin the whole lesson over, or to go back to where you began working with the Text Editor, type **CTRL-T**.

If you still need help remembering the function of certain commands, press **ESC H RETURN** to display all the simple Text Editor commands and their functions; press the **RETURN** key to return you to the point in the file where you left off.

PIE Command Key	Function
ESC H RETURN	Displays all simple Text Editor commands and their functions; press return to continue

IMPORTANT NOTE: This manual documents two sets of lessons. If you are using an Apple II, Apple II Plus, or Franklin ACE, then refer to the first set of lessons for the Standard Keyboard.

If you are using an Apple //e *and* have selected option 1 in the System Configure program to use the Apple //e arrows keys for cursor movement, then refer to the second set of lessons for the Apple //e Keyboard. However if you selected option 2 in the System Configure program to use the Apple II Plus keyboard layout, then you want to refer to the first set of lessons for the Standard Keyboard.

PIE TEXT EDITOR LESSONS FOR THE STANDARD KEYBOARD

Lesson 1: The Cursor

When the first Text Editor window appears on your screen, the cursor flashes in the top left-hand corner of the window.

When you want to scroll forward to the next page, use the **CTRL-V** (page down) command. The **CTRL-V** command advances the screen and cursor 21 lines, or 1 page, at a time.

PIE Command Key	Simple Function
CTRL-V	Advances screen 21 lines (1 page) at a time

Lesson 2: Home and Bottom Home

The second lesson describes the function of the **CTRL-D** (home) command. **CTRL-D** is a toggle command that moves the cursor to the top left-hand and the bottom left-hand corner of the screen.

Press **CTRL-D** to send the cursor Home from any cursor position on the screen. Press **CTRL-D** again to send the cursor to Bottom Home.

PIE Command Key	Simple Function
CTRL-D	Homes the cursor at the top left-hand or bottom left-hand corner of the screen (toggle)

Lesson 3: Simple Cursor Movement

The simplest cursor movements (up, down, left, and right one space) have been positioned on your keyboard so that you can perform them quite easily with one hand. These commands—**CTRL-E** (up), **CTRL-C** (down), **CTRL-S** (left), and **CTRL-F** (right)—combine with **CTRL-D** to form a lopsided cross on the keyboard.

Notice that when you move the cursor with these commands, the text the cursor passes over remains intact.

PIE Command Key	Simple Function
CTRL-E	Moves the cursor up 1 line
CTRL-C	Moves the cursor down 1 line
CTRL-S	Moves the cursor 1 space to the left
CTRL-F	Moves the cursor 1 space to the right

Lesson 4: Cursor Movement Practice

Practice the cursor commands you have just learned. These commands are probably used more often than any other group of commands, so it is important to become familiar with them.

The **REPT** key on the Apple II and II Plus repeats a single command as many

times as you want, for as long as you hold down the **REPT** key and the command sequence at the same time. Use the **REPT** key to make the cursor move up, down, left, or right faster.

PIE Command Key	Simple Function
REPT-	Repeats any command for as long as you hold down the **REPT** key and the command at the same time

The Apple //e uses automatic repeat. As long as you hold any key down, the character will be automatically repeated.

Lesson 5: The Status Line

The Status Line is displayed just below the bottom border of the window on the screen. The Status Line displays information about the cursor position and the current text entry "mode."

The Status Line also keeps track of other details, such as INSERT MODE, and displays them on the screen. Keep an eye on the Status Line as you work.

Lesson 6: Beyond the Right-Hand Margin

The PIE Writer screen can display only 38 characters horizontally at a time {78 in the 80-column version}. There is additional typing room beyond the right-hand margin, though.

A right caret (>) on the right-hand side of the main page indicates that there is text beyond the right-hand margin on that line. Use the **CTRL-F** command to move the cursor to the right until the right-hand side of the page appears.

As you type the 39th {or 79th} character in the line, you will hear the bell column tone. The bell column is the 38th {or 78th} column. It is displayed as a reverse video dash at the top right-hand corner of the main page window. This setting is a system default. You can set the bell tone in any column you want, or clear the bell column completely, with an advanced command (introduced in Chapter 5).

Press **RETURN** to get back to the main page.

Lesson 7: Tabs

You can move back and forth across the page quickly by using the tab commands, **CTRL-G** (tab right) and **CTRL-A** (tab left). **CTRL-G** moves the cursor one tab stop to the right; **CTRL-A** moves the cursor one tab stop to the left.

Each tab stop is indicated by a plus (+) sign at the top of the window. These settings are a system default. You can change the position and the number of tab stops with more advanced tab commands (introduced in Chapter 5).

If you tab the cursor beyond the right-hand margin, you can get back to the main page by tabbing left with **CTRL-A** or by pressing **RETURN**.

PIE Command Key	Simple Function
CTRL-G	Moves the cursor 1 tab stop to the right
CTRL-A	Moves the cursor 1 tab stop to the left

Lesson 8: Moving the Cursor to the Beginning or End of a Line

The **CTRL-B** command, like **CTRL-D**, toggles back and forth between two different functions. Press **CTRL-B** once to move the cursor to the end of the line, even if the end of the line is beyond the right-hand margin. Press **CTRL-B** again to move the cursor back to the left margin.

The **CTRL-B** command makes it easy to add text at the end of a line or begin a new sentence there.

PIE Command Key	Simple Function
CTRL-B	Moves the cursor to the beginning or end of a line (toggle)

Lesson 9: Scrolling Pages and Lines

You have been taught how to scroll the entire page forward using **CTRL-V** (page down). You also can scroll the page up or down a screen or a line at a time.

CTRL-R (page up) lets you scroll back a page at a time to review text.

You also can scroll up a single line at a time with **CTRL-Y** and down a line at a time with **CTRL-N**.

PIE Command Key	Simple Function
CTRL-V	Advances screen 21 lines (1 "page") at a time

CTRL-R	Reverses screen 21 lines (1 "page") at a time
CTRL-Y	Reverses screen and cursor 1 line at a time
CTRL-N	Advances screen and cursor 1 line at a time

Lesson 10: Upper and Lowercase Character Display

This lesson is only for Apple II and Apple II Plus owners who are using a 40-column version of PIE Writer and have *not* installed a lowercase adapter. All other readers can skip this lesson because their video screens display true upper and lowercase characters. (Standard Apple II and Apple II Plus systems do not.)

The standard Apple II screen displays all characters in uppercase. It distinguishes between the upper and lowercase characters you enter at the keyboard, though, by showing uppercase in reverse video (black letters on white background) and lowercase normally (white letters on a black screen).

You enter uppercase characters with → key or **SHIFT** key, as explained below.

Lesson 11: Uppercase Text Entry

The **SHIFT** keys on a standard Apple II and Apple II Plus computer do not capitalize lowercase characters as they do on a standard typewriter, unless you have installed the "Shift Key Mod" described in Chapter 6. So the → key lets you enter uppercase letters on the screen. Press → once to enter the next character you type in uppercase. You are returned to lowercase entry mode immediately after entering the single upper case character.

The **SHIFT** key on the standard Apple II and II Plus is only used to enter upper-register symbols and punctuation marks. Hold down the **SHIFT** key while typing one of the Apple's upper-register characters, which are: ! " # $ % & ' ( ) * = @ + < > and ?.

If you have installed the "Shift Key Mod", or are using an Apple //e or Franklin ACE computer, then you may also use the **SHIFT** keys to enter uppercase characters on the keyboard.

The Apple II and II Plus do not have a **CAPS LOCK** key. Press → twice to lock in uppercase. Then enter as many uppercase characters as you want. You are not returned to lowercase entry mode until you press → twice a second time.

Both the Apple //e and Franklin ACE have a **CAPS LOCK** key. When depressed (locked), all characters entered will be in uppercase. Press it again to unlock it.

PIE Command Key	Simple Function
→ **or** **SHIFT**	Makes next character entered uppercase
→ → **or** **CAPS LOCK**	Locks in and unlocks uppercase mode (toggle)

Lesson 12: Entering Upper and Lowercase Characters

Enter one or more letters in uppercase, in combination with lowercase letters. Type all over the screen. Practice using the → key or **SHIFT** and **CAPS LOCK** keys to learn easy upper and lowercase character entry.

If you shift right after typing the 38th {or 78th} character in a line, press **RETURN** to get back to the main page.

Lesson 13: Destructive Cursor Moves

The first twelve Text Editor lessons cover cursor control, window control, and text entry. Lessons 13 through 16 teach simple editing commands.

The **space** bar and ← (backspace) key each perform a simple cursor movement, but they also erase whatever text characters are in their path.

The **space** bar deletes a character, then moves the cursor 1 space to the right of where that character was.

The ← key deletes a character to the left as it backspaces.

Note that when deleting the characters on the screen, only the lesson in memory changes—but the lesson file is still stored on the disk intact for you to use again.

PIE Command Key	Simple Function
space	Deletes a character and moves the cursor 1 space to the right
←	Deletes a character to the left as it backspaces

Lesson 14: Deleting a Character

The **CTRL-J** (delete) key deletes a character or a blank and shifts the rest of a line left toward the cursor. Position the cursor on any letter you want to delete, and type

CTRL-J to edit a misspelled word.

PIE Command Key	Simple Function
CTRL-J	Deletes a character at the cursor and moves the other characters on the line 1 space to the left

Lesson 15: Inserting a Character

The **CTRL-P** (insert) command allows you to enter characters anywhere within a word or a line.

CTRL-P toggles between entering and leaving the insert mode. Press **CTRL-P** to enter the insert mode, then enter as many characters in the middle of a word or a line as you want. All the characters on the line, beginning with the character at the cursor position when you first enter the insert mode, shift right as you enter text.

Press **CTRL-P** again to leave the insert mode.

PIE Command Key	Simple Function
CTRL-P	Begins or ends the insertion of characters at the cursor

Lesson 16: Shifting Text Left or Right

You can shift a line of text left or right while you are in the insert mode.

First, type **CTRL-P** to enter the insert mode.

Now, to shift a line to the right, press the **space** bar. All text on the line at or to the right of the cursor position before you entered the insert mode shifts right.

To shift a line to the left, press the ← (backspace) key. The ← key lets you "drag" a line to the left.

The editing commands taught in these last few lessons should make one thing clear: You need not enter text perfectly the first time with PIE Writer. You can always review and edit your document later.

PIE Command Key	Simple Function
CTRL-P space	Shifts a line of text to the right 1 space
CTRL-P ←	Shifts a line of text to the left 1 space

Lesson 17: Returning to the Beginning of a File; Leaving the Text Editor and Entering the Command Processor

Type **CTRL-T** to return to the first page of your file. **CTRL-T** takes you to the beginning of any file created with the Text Editor.

Remember that you can scroll backward or forward a page or screen at a time. To go back to a specific place in your file, use **CTRL-R** (page up), **CTRL-V** (page down), **CTRL-Y** (scroll up), and **CTRL-N** (scroll down).

When you have finished working with the Text Editor, type **CTRL-SHIFT-P** {Apple //e and Franklin: use **CTRL-@**} to leave the Text Editor and enter the Command Processor.

PIE Command Key	Simple Function
CTRL-T	Moves the cursor to the beginning of a file
CTRL-SHIFT-P or **CTRL-@**	Leave the Text Editor and enter the Command Processor

The manual documents more sophisticated and powerful word processing commands that can be used by any PIE Writer user. All these commands are introduced in Chapter 5.

When you leave the Text Editor with **CTRL-SHIFT-P** {Apple //e and Franklin: **CTRL-@**}, you enter the Command Processor.

THE COMMAND PROCESSOR

The Command Processor (CP) manages all the operating files on the system diskette and all the document files you create using the Text Editor. Some of the functions of the CP are:

1. *Loading a file from the diskette so you can review, edit or add to it.*
2. *Saving text you have written on a diskette file and labeling the file with a name you specify.*
3. *Clearing the memory so that you can begin a brand new file.*
4. *Reminding you of the filename of the text you are working on.*
5. *Reporting on the number of words or lines in a file.*

5

Reference Section

This section contains explanations of every PIE Text Editor, PIE Command Processor, and FORMAT Text Processor command in the PIE Writer system. The basic commands have already been covered in earlier sections; however, new, advanced commands are presented here for the first time. The section ends with a discussion of error messages you may encounter. An abbreviated list of all PIE Writer commands is provided on the Reference Card.

PIE TEXT EDITOR

PIE Text Editor commands are documented here in groups according to the similarity of their functions.

Within each group you will find both simple commands and compound commands described. Compound commands are formed by preceding the single-key simple commands with prefixes. All compound commands are initiated by first typing the **ESC** key; some compound commands also take a modifier, which is typed after the **ESC** key is pressed.

After **ESC** is pressed, the word "ARG:" appears on the lower right of the Status Line. ("ARG" is short for "argument".) Sometimes simply prefixing a simple command with the **ESC** key is enough to change the function of the command; just as often, arguments or modifiers are entered. The text of the modifier appears on the status line as you enter it in.

Possible modifiers include: a specific letter, some arbitrary character (indicated by *c* in the descriptions below), a number (indicated as *nn* or *mm*), or a string of characters (represented by *s*, *s1* or *s2*). The modifiers and the simple commands which they prefix are shown in regular type; the simple commands themselves are in bold face.

Simple Cursor Movement

The cursor indicates where your next typed character will appear. The cursor's column and line position are reflected in the Status Line at the bottom of the screen. With the following commands you can move the cursor to any position on the screen.

PIE Command Key	Function
CTRL-C	Moves cursor down one line
CTRL-E	Moves cursor up one line
CTRL-F	Moves cursor one space to the right
CTRL-S	Moves cursor one space to the left

Each of the vertical cursor movement commands triggers single-line scrolling if the cursor movement would otherwise cause the cursor the leave the text window.

Cursor Jumps

PIE Command Key		Function
	CTRL-D	Toggles between moving cursor to the top left or bottom left corner of screen
	CTRL-B	Toggles between moving cursor to beginning or end of line
ESC nn	CTRL-F	Moves cursor to column 'nn'
	RETURN	Moves cursor to column one of next line

CTRL-D alternates between one of two functions: it toggles back and forth between moving the cursor to the top left-hand corner and moving it to the bottom left-hand corner of the screen. From any cursor position, other than the top left-hand corner,

CTRL-D moves first to the top left-hand corner.

CTRL-B initially moves the cursor to the end of whatever line the cursor is on, then toggles back and forth between moving the cursor to the beginning and to the end of that line.

ESC nn CTRL-F moves the cursor to column 'nn'. Values from 1 to 64 are valid {1 to 128 for 80-column versions}.

The effect of the **RETURN** key depends on the current text entry made (Manual, PPWrap, or Indent).

In Manual mode, the **RETURN** key always performs only a cursor movement: it causes the cursor to go to the beginning of the next line. In PPwrap or Indent modes, typing **RETURN** at the end of a line inserts a blank line after the current one, and then moves the cursor to the position just below the leftmost character of the current line.

Tabbing

PIE Command Key		Function
	CTRL-G	Moves cursor one tab stop to the right
ESC S	CTRL-G	Sets bell column to current column
ESC C	CTRL-G	Clears bell column
	CTRL-A	Moves cursor one tab stop to the left
ESC S	CTRL-A	Sets tab marker in current column
ESC C	CTRL-A	Clears tab marker in current column
ESC 0	CTRL-A	Clears all tab markers
ESC nn	CTRL-A	Sets tab markers every 'nn' columns
ESC W	RETURN	Toggles between word and column tabbing

Attempting to tab beyond the edges of the screen has no effect, except in Word Tab mode. In Word Tab mode, instead of tabbing to the next tab stop, you can tab to the first character of the next word to the right or left: **CTRL-G** tabs to the word on the right, and **CTRL-A** tabs to the word on the left. If there are no more words on the line, a tab command simply moves to the first letter of the last word on the line above (**CTRL-A**) or to the first letter of the first word on the line below (**CTRL-G**).

ESC S CTRL-G sets the bell column at whatever column the cursor is in. **ESC C CTRL-G** clears the bell column, regardless of the cursor position and the column in which the bell is set. The bell column is indicated by a reverse video character {or vertical bar} at the top of the screen; it functions like the bell on a typewriter, causing the computer to "beep" when the cursor moves past it.

ESC S CTRL-A sets a tab at the column the cursor is in; **ESC C CTRL-A** clears a tab at the cursor. All commands which change the tab stops are effective whether or not you are in column tabbing mode. **ESC 0 CTRL-A** clears all tabs, regardless of the cursor position and the number of columns in which tabs are set. **ESC nn CTRL-A** sets a tab every 'nn' columns beginning at the first column; valid values for 'nn' are from 1 to 64 {1 to 128 for 80-column versions}.

The type of tabbing (word or column) is set separately for each of the three text entry modes (Manual, PPwrap, and Indent). Initial default settings are Word Tabbing for PPWrap mode and column tabbing for Manual and Indent modes. **ESC W RETURN** toggles the tabbing method for the current text mode. If you shift between modes, the type of tabbing previously selected in that mode is restored.

Line and Window Scrolling

	PIE Command Key	Function
	CTRL-V	Advances screen 21 lines (one page)
ESC nn	CTRL-V	Advances 'nn' pages
ESC	CTRL-V	Moves current line to top of screen
	CTRL-R	Reverses screen 21 lines (one page)
ESC nn	CTRL-R	Reverses 'nn' pages

	CTRL-N	Advances cursor and screen one line
ESC	CTRL-N	Advances cursor and screen one-half page
ESC nn	CTRL-N	Advances cursor and screen 'nn' lines
	CTRL-Y	Reverses cursor and screen one line
ESC	CTRL-Y	Reverses cursor and screen one-half page
ESC nn	CTRL-Y	Reverses cursor and screen 'nn' lines

CTRL-V advances the screen 21 lines, or one "page", and **CTRL-R** reverses the screen 21 lines. The cursor remains in its original position on the screen.

ESC nn CTRL-V advances the screen 'nn' pages; **ESC nn CTRL-R** reverses the screen 'nn' pages.

ESC CTRL-V moves the line the cursor is in to the top of the screen, homing the line. The cursor remains in the same column.

CTRL-N advances the screen one line. The cursor advances with the line it was on. **CTRL-Y** reverses the screen and cursor one line.

ESC nn CTRL-N advances the screen and cursor 'nn' lines, and **ESC nn CTRL-Y** reverses the screen and cursor 'nn' lines. **ESC CTRL-N** advances the screen and cursor one half page, and **ESC CTRL-Y** reverses the screen and cursor one half page.

Go To a Line

PIE Command Key		Function
	CTRL-T	Go to beginning of file
ESC	CTRL-T	Go to end of file
ESC nn	CTRL-T	Go to line 'nn'

CTRL-T moves the cursor and screen to the beginning of the file you are working on; **ESC CTRL-T** moves the cursor and screen to the end of that file. **ESC nn CTRL-T** moves the cursor and screen to line 'nn'.

Some commands, such as go to a line, take a few seconds to complete. PIE Writer displays a "BUSY" message on the status line during this time. Even while the "BUSY" message is being displayed, however, you can continue entering commands (or text). This is called 'type-ahead'; the computer remembers what you have typed and enters it at the cursor's original position after the first command has been carried out.

Inserting and Deleting

	PIE Command Key	Function
	CTRL-P	Begins or ends insertion mode
ESC s	CTRL-P	Inserts string 's' at cursor
	←	Deletes character to the left and backspaces
	CTRL-I	Inserts a blank line at cursor
ESC nn	CTRL-I	Inserts 'nn' blank lines at cursor
	CTRL-J	Deletes character at cursor and moves other characters one space to the left
ESC c	CTRL-J	Deletes all characters up to but not including 'c'
ESC	CTRL-J	Deletes all characters to the right of the cursor
	CTRL-SHIFT-N or **CTRL-^**	Deletes word at cursor

ESC	CTRL-SHIFT-N or CTRL-^	Deletes line at cursor
ESC nn	CTRL-SHIFT-N or CTRL-^	Deletes 'nn' lines at cursor
ESC #mm,nn	CTRL-SHIFT-N or CTRL-^	Deletes lines 'mm' through 'nn'
ESC	CTRL-B	Deletes all blank lines below cursor

CTRL-P toggles in and out of the insert mode, in which a string of characters can be inserted at the cursor. The message "INSERT MODE" appears on the Status Line when the insert mode is active; it disappears when **CTRL-P** toggles out of it. If there is not enough space on the line for character insertion while in the insert mode, the error message "NOT ENOUGH ROOM ON LINE" will be displayed on the Status Line.

ESC s CTRL-P allows you to insert 's'—any string of characters—at the cursor, forcing all other text on the line to make room by moving to the right. **ESC CTRL-P** inserts 's'—that is, the same string—at the cursor again. The string is limited to 26 {or 58 in 80-column versions} characters, including spaces.

The ← key deletes text as it backspaces. This command has no effect if the cursor is in column one. In insert mode (initiated by **CTRL-P**), ← drags every character to the right of the cursor to the left as it deletes and backspaces.

CTRL-I inserts a blank line, and **ESC nn CTRL-I** inserts 'nn' blank lines, at the cursor. The line the cursor was on before the command moves below the inserted line.

The **CTRL-J** command deletes a character at the cursor and shifts all the text to the right of the cursor to the left. **ESC c CTRL-J** deletes all the characters on a line from the cursor to the first occurrence of 'c' on the same line; 'c' may be any character. The text to the right of the deletion shifts left. **ESC CTRL-J** deletes every character to the end of a line from the cursor.

CTRL-SHIFT-N {or **CTRL-^** on the Apple //e} deletes an entire word regardless of the cursor's position within that word. The text to right of the deleted word shifts left. If the cursor is on a space preceding a word, all spaces from the cursor

to the start of the word are deleted. In PPWrap mode, if the word deleted is the only word on a line, a line delete is performed, closing up the created space between the lines above and below.

ESC CTRL-SHIFT-N {or **CTRL-^**} deletes an entire line regardless of the cursor's position within the line, and all subsequent lines are moved up. **ESC nn CTRL-SHIFT-N** {or **CTRL-^**} deletes 'nn' lines at and beneath the cursor.

ESC #mm,nn CTRL-SHIFT-N {or **CTRL-^**} deletes lines 'mm' to 'nn', including lines 'mm' and 'nn.' {**In the 48K version, this command requires the loading of code from the PIE AUX file so the PIE Writer system disk must be in the program drive when it is executed.**} This command also takes several seconds to execute, so the Status Line will show BUSY while the command is being carried out; type-ahead may be used.

A '$' may be used instead of a number for 'nn' to stand for the last line in the file.

Splitting and Joining Lines

	PIE Command Key	Function
ESC	CTRL-I	Splits a line at the cursor
ESC	CTRL-K	Joins two lines together

ESC CTRL-I splits a line at the cursor and moves whatever text was to the right of the cursor to the left margin of the new line.

ESC CTRL-K joins a line below the cursor to the end of the line the cursor is on. The cursor must be positioned following the final character in a line, or else the Status Line says "CLEAR TO EOL (end of line) FIRST." If the line below the cursor is too long to be appended to the current line, the message "NOT ENOUGH ROOM ON LINE" is displayed on the Status Line.

Copying and Moving Lines

The ability to copy lines into memory and move them to different positions in your text is a powerful editing function. Using the copy functions, any amount of text may be copied from one position to another position within a file. Information held in the copy buffer remains there until another move or copy command is given, and so may be recalled repeatedly. It even remains intact after loading in a new file so that short blocks of text may be moved between files easily.

	PIE Command Key	Function
	CTRL-L	Copies line at cursor into memory
ESC nn	CTRL-L	Copies 'nn' lines into memory
	CTRL-K	Copies line at cursor into memory and deletes it from screen
ESC nn	CTRL-K	Copies 'nn' lines into memory and deletes them from screen
	CTRL-O	Recalls lines from memory and copies them onto the screen at cursor
ESC #mm,nn	CTRL-O	Moves lines 'mm' to 'nn' to current cursor position

CTRL-L copies a single line of text into memory. The Status Line blinks once while the line at the cursor is being copied. **ESC nn CTRL-L** copies 'nn' lines beginning at the cursor into memory, up to a maximum of 21 lines of text, or one full window.

The **CTRL-K** command copies a single line of text at the cursor into memory and deletes it from the screen. This is the safest method you can use for deleting a line of text, because it can always be recalled using **CTRL-O**. You can save a copy of up to one window, or 21 lines, of text with **ESC nn CTRL-K**, which saves 'nn' lines in memory while deleting them from the screen.

The **CTRL-O** command allows you to recall the set of lines placed in memory with either **CTRL-L** (which copied them into memory) or **CTRL-K** (which copied and deleted). The text recalled is inserted at the current cursor position. The cursor may be repositioned and the **CTRL-O** command may be used again to insert as many copies of the block of lines as you like.

ESC #mm,nn CTRL-O moves a range of any number of lines. Move the cursor to the place in the text where you want to move lines 'mm' to 'nn.' Enter the **ESC #mm,nn CTRL-O** command, and the range of lines 'mm' to 'nn' will be retrieved from memory and placed at the cursor. {For the 48K version, the system diskette with the PIE AUX file must be in the program drive to execute this command.}

Again, '$' may be used to represent the end of file.

Search and Replace

A "string" is a sequence of characters. A string might be a sentence, a word, or a single character. PIE limits search strings to a maximum of 26 {58 in 80-column versions} characters. The next set of commands instructs PIE to search for a string and to position the cursor at the string when it has been found. Failed searches produce a "SEARCH KEY NOT FOUND" message on the Status Line. (The word "KEY" refers to the search string.)

	PIE Command Key	Function
	CTRL-Z	Searches forward for current search string
ESC s	CTRL-Z	Sets search string to 's' and searches forward
	CTRL-Q	Searches backwards for current search string
ESC s	CTRL-Q	Sets search string to 's' and searches backward

To find a string, type **ESC** to display the ARG: prompt on the Status Line. Enter the string ('s') you want found. After the string is entered, press **CTRL-Z** to search toward the end of the file or press **CTRL-Q** to search toward the beginning of the file. PIE Writer remembers the search string until you type in a new one. To search for the next occurrence of the same string, press **CTRL-Z** to search forward again, or **CTRL-Q** to search backward again.

If the string exists but is split across two lines it will not be found.

A search string can contain ambiguous or "wild card" characters (except for the first character). The wild card character matches any letter. The ASCII DEL or rubout character is the wild card, and is entered by typing → **3** or → **#**. (See "Entering Control and Nonkeyboard Charactrers" for more on entering nonkeyboard characters.)

For instance, a string consisting of the letter "C," the letter "A," and the DEL character will match "CAT," "CAP," and "CAR" during a search.

	PIE Command Key	Function
	CTRL-X	Replaces next occurrence of search string with replace string
ESC s	CTRL-X	Sets replace string to 's', then replaces next occurrence of search string with 's'
ESC s1 ESC s2	CTRL-X	Sets search string to 's1', replace string to 's2', and replaces next occurrence of 's1' with 's2'
	CTRL-W CTRL-X	Replaces all occurrences of search string with replace string
ESC s1 ESC s2	CTRL-W CTRL-X	Sets search string to 's1', replace string to 's2', and replaces all occurrences of 's1' with 's2'

The search and replace commands make use of two different strings, a 'search' string and a 'replace' string, each of which is stored in a separate place in memory ('s1' refers to the search string and 's2' to the replacement string). As can be seen above, there are several ways in which these two strings may be entered into memory and new values given to the the search and replace commands.

ESC s1 ESC s2 CTRL-X searches forward for a string ('s1') and replaces it with another string ('s2'). Pressing **CTRL-Z** (or **CTRL-Q**) will then search forward (or backwards) to find the next string that matches the search string. Typing **CTRL-X** performs another replacement at the next occurrence of the search string; that is, it does an automatic forward search. To enter a different replacement string but leave the search string intact, use **ESC s CTRL-X**.

If the replacement string ('s2') causes the line to exceed the available space on the line, the command will not execute, and the Status Line will display "NOT ENOUGH ROOM ON LINE". If the search string cannot be found, then the replacement cannot be made, so the Status Line displays "SEARCH KEY NOT FOUND". Remember that

CTRL-X does only a forward search.

To change every occurrence of a string throughout the text, use **ESC s1 ESC s2 CTRL-W CTRL-X**. This looks for the search string ('s1') and replaces it with the replacement string ('s2') everywhere in the file. This function is called a "global" search and replace.

Every occurrence of the search string will be replaced, unless a line is made too long by the replacement string, in which case the command quits. You may edit the line, the press **CTRL-W CTRL-X** again, and the global search and replace continues forward searching and replacing using the same strings.

The sequence **CTRL-W CTRL-X** takes the most recent search and replacement strings and performs a global search and replace from the cursor position to the end of the file.

Note that using **ESC s CTRL-P** to make an insertion changes the current replacement string.

Text Entry Modes

	PIE Command Key	Function
ESC	RETURN	Toggles PPwrap with Manual mode
ESC I	RETURN	Enters Indent mode

A standard typewriter requires a carriage return at the end of every typed line. PIE Writer's Manual mode works the same way: the **RETURN** key must be pressed to advance the cursor to the beginning of the next line. Manual mode is what you are in when you first enter the Text Editor.

ESC RETURN (or **ESC P RETURN**) sets the PPWrap mode from the default Manual mode. PPWrap mode allows you to type continuously without pressing the **RETURN** key: whenever a character is typed beyond the bell column, a blank line is inserted below the current line and all of previous characters in the same word are automatically moved down. The entire word wraps around the screen and begins at the beginning of the new line. A new line is also inserted if **RETURN** is pressed when the cursor is at the end of a line.

ESC I RETURN lets you enter the Indent mode. In this mode, pressing the **RETURN** key moves the cursor to the position below the leftmost charcter on the previous line. This is useful for entering outlines, lists, and program text, such as Pascal or Assembly code.

```
In the Indent mode
    a line indented this much
    is indented again automatically
    after every carriage return.
```

You must press **RETURN** to perform a carriage return in the Indent mode. Text does not wrap around the screen automatically. Pressing **RETURN** does make room for any new text you may want to enter, however, by forcing all text below the cursor down one line.

ESC RETURN from either PPWrap or Indent mode returns you to Manual mode.

Displaying the Help Screen

PIE Command Key		Function
ESC H	RETURN	Displays Help screen

ESC H RETURN displays the Help screen, a listing of all the simple functions and their command keys. However, this command is available only if the Help file already has been loaded from the Command Processor by typing **%H**. (Refer to the section on the Command Processor in this chapter.) Otherwise the message "INVALID PARAMETER" is output.

Using PIE with the Help screen loaded reduces the size of your edit buffer by 1024 characters. It is intended to be used while learning the Text Editor commands.

By pressing **RETURN** while viewing the Help screen, you are returned to the point in the file where you left off.

Displaying and Entering Upper-and Lowercase Letters

An unmodified Apple II can only display uppercase letters on the screen. It differentiates between capital and lowercase letters by showing black characters on a white background (reverse video) for uppercase and white characters on a black background for lowercase letters. This way, the relatively few capital letters stand out from the majority of lower case letters in normal text entry.

Two methods are available for displaying true upper-and lowercase letters for the Apple II and II Plus. Adapters are sold by several manufacturers that allow upper and lowercase display on standard Apple II systems. A PIE Configure option (Chapter 4) notifies PIE Writer that you have added this feature to your system for true character display.

Also, 80-column display boards include true upper and lowercase display capabilities.

PIE Command Key	Function
→	Makes next character entered uppercase

The **SHIFT** keys on the standard Apple II and II Plus computers do not generate uppercase letters as they do on a typewriter. Using the → key lets you enter the next character in uppercase. Pressing the → twice locks in uppercase entry, so that you can type in strings of capital letters without continually pressing the → prior to each character typed. Pressing the → twice more returns you to lowercase entry mode.

On the Apple II and II Plus, whenever PIE Writer is in the lowercase mode, the Status Line displays "LOCASE" {"LOWER CASE" in 80-column versions}. Absence of "LOCASE" on the Status Line indicates that you are in uppercase entry mode.

Both the Apple //e and Franklin have true upper and lowercase display. They also have **SHIFT** keys that generate uppercase characters, and a **CAPS LOCK** key which locks all characters into uppercase while it is depressed. There is no indication on the screen of the status of the **CAPS LOCK** key.

In Chapter 6, a simple hardware modification is described that makes the **SHIFT** key on the Apple II and II Plus perform the way it does on a regular typewriter.

Contact your dealer for additional information on lowercase adapters, 80-column display boards, and the **SHIFT** key modification.

Cursor-Defined Commands

With cursor-defined commands, you allow the horizontal and vertical motion of the cursor to define the area of the screen you want a command to be effective in, so that the same command can be applied to more than one line of text at a time. Cursor-defined commands can be carried out only for on-screen text below or to the right of the cursor's position before the cursor-defined command is entered.

Cursor defined commands allow you to count lines and columns, using cursor movement, to specify the range of a command, just as a numeric value can be used to count those same lines and columns. For example, **ESC 8 CTRL-I** inserts 8 blank lines at the cursor.

The cursor-defined analog of this command is described below.

	PIE Command Key	Function
ESC ⇨⇩	CTRL-I	Block move right
ESC ⇨⇩	CTRL-K	Block move left
ESC ⇨⇩	CTRL-SHIFT-N or CTRL ^	Block move left with delete
ESC ⇩	CTRL-L	Copy lines to memory
ESC ⇩	CTRL-K	Delete lines to memory
ESC ⇩	CTRL-SHIFT-N or CTRL ^	Delete lines

The symbols ⇩ and ⇨ are used to represent vertical and horizontal cursor movement. Commands which only have the ⇩ symbol only allow vertical movement; addition of the ⇨ means that the command uses horizontal cursor movement as well.

Cursor-defined modifiers are entered by first typing **ESC**, then entering one of the seven simple cursor movement commands noted below. As soon as a cursor command is typed following the **ESC**, the original cursor position is replaced on the screen with the @ symbol, and the message "CURSOR DEFINED" is displayed on the Status Line. For vertical cursor-defined commands, the net cursor movement must be down; upwards cursor movement is allowed, but not above the original cursor position. The cursor movement is used indicate the range over which the command is effective; the original cursor position serves as one end point, the final cursor position as the other.

Subject to this restricition, the simple commands **CTRL-C** and **CTRL-E** function as usual to move the cursor vertically. The **CTRL-D** cursor move, however, works differently. Instead of toggling between Home and Bottom Home, as a modifier, it toggles between the original cursor line and the bottom line on the screen, remaining in the current column.

The horizontal cursor-defined commands use the **CTRL-F**, **CTRL-S**, **CTRL-G**, and **CTRL-A** simple commands to define the horizontal range over which a command is effective. Similarly to the vertical cursor-defined commands, the net cursor movement must be to the right of the original cursor position.

All of the horizontal cursor-defined commands can also be used with vertical cursor movement to define the range of lines over which the horizontal function is to be performed; thus "blocks" of text on the screen may be shifted.

ESC ⇨⇩ CTRL-I performs a cursor-defined block move right. Suppose that the cursor is positioned on the 'T' in the word 'this' below, and you want to shift two lines of text so that they begin eight columns to the right.

```
THIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

Press **ESC** and then press **CTRL-G**. The Status Line displays the message "CURSOR DEFINED".

The cursor's original position when you entered the command is marked with an @ symbol on the screen.

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

Assuming your tab stops are set to 8, the **ESC CTRL-G** tabs the cursor to the 'A' in 'AN':

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

CTRL-C moves the cursor down one line so that the cursor is positioned over the 'S' in 'USE' on the second line:

```
@HIS IS AN EXAMPLE OF
HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

CTRL-I inserts spaces from @ to the current cursor position:

```
        THIS IS AN EXAMPLE OF
        HOW TO USE HORIZONTAL AND
VERTICAL CURSOR-DEFINED COMMANDS
```

ESC ⇨⇩ CTRL-K performs a cursor-defined block move left. The **CTRL-K** command is preceded by a simple horizontal command (to indicate the number of spaces you want to shift the text left) and a simple vertical command (to indicate how many lines you want shifted). Note that the cursor movement is in the direction *opposite* that of the text shift. In the example below, you can shift the block of text to the left by first positioning the cursor in the column where you want the first letter of the line to end up, then pressing **ESC**:

```
@    THIS IS ANOTHER EXAMPLE OF
     HOW TO USE HORIZONTAL AND
     VERTICAL CURSOR-DEFINED COMMANDS
```

Move the cursor so that it is positioned over any part of the text in the first line with **CTRL-F** or **CTRL-G** commands, then use a **CTRL-C** to move the cursor onto the line below it. Type a **CTRL-K** command to see the screen below:

```
THIS IS ANOTHER EXAMPLE OF
HOW TO USE HORIZONTAL AND
    VERTICAL CURSOR-DEFINED COMMANDS
```

ESC ⇨⇩ CTRL-SHIFT-N shifts text to the left, but also deletes characters as defined by the cursor movement. In the example above, the cursor would have had to have been positioned horizontally on the 'T' in 'THIS,' then moved to the 'H' in 'HOW' with a **CTRL-C** to get the same result. Positioning the cursor anywhere within the lines would have deleted all the characters to the left of the cursor, then shifted the remaining characters on the lines left to the column where the @ was first set.

ESC ⇩ CTRL-L copies lines into memory as defined by the cursor movement; it functions identically to **ESC nn CTRL-L**, with 'nn' simply being defined by the cursor movement rather than with a number. The entire lines are placed in memory regardless of the column the cursor is in.

ESC ⇩ CTRL-K places lines into memory as defined by the cursor movement and deletes them from the screen; again, it is similar to **ESC nn CTRL-K**.

ESC ⇩ CTRL-SHIFT-N deletes lines as defined by the cursor movement, and the lines below those deleted close up the space created.

Entering Control and Nonkeyboard Characters

Control characters cannot be entered into a text file directly using the Text Editor for the simple reason that all of the keyboard control characters are interpreted as Text

Editor commands. Naturally, it is desirable to be able to enter any ASCII character into your file. (A typical application would be to imbed printer commands in your file.)

PIE Command Key	Function
CTRL-SHIFT-M or **CTRL-]**	Allows entry of next character as a control code

CTRL-SHIFT-M lets you enter any control character literally, that is, without interpretation. Executing a **CTRL-SHIFT-M** produces the message "ENTER CTRL CHAR" on the Status Line.

For control characters in the range CTRL-@ to CTRL-Z, simply type the corresponding key; you do not need to press **CTRL**. For control characters not available on the Apple keyboard, (like CTRL-^), enter the appropriate translation character. (Refer to the ASCII Reference Chart in Appendix A for the right translation characters.) The control character you enter appears on the screen as a flashing character. {On 80-column versions, it depends on your board, but they are usually displayed in inverse video.}

Most printing characters are entered simply by typing the keys (the → key is normally used to toggle between uppercase and lowercase, but the **SHIFT** key can be used if the hardware modification has been made).

Keys unavailable on the Apple keyboard can be entered using the → key followed by the non-alpha key listed under "Translation Character" in the ASCII Reference Chart. For machines without lowercase adapters or 80-column boards, these will be displayed by indicated, different characters.

PIE Command Key	Function
→	Allows entry of non-keyboard characters by translating 'n'

The special characters, ^, [, and @ can only be entered using the **SHIFT** key when in uppercase lock mode (uppercase lock corresponds to the standard Apple keyboard). These may be entered from the lowercase mode using the translation characters.

Again, the ASCII Reference chart in Appendix A indicates the keystrokes necessary to enter these nonkeyboard characters.

Exiting the Editor and Transferring to the Command Processor

The final Text Editor command transfers you to the Command Processor (CP).

PIE Command Key	Function
CTRL-SHIFT-P or **CTRL-@**	Exits Editor and returns to Command Processor

You exit all PIE Text Editor files and transfer to the PIE Command Processor by executing a **CTRL-SHIFT-P**. Your text remains intact in the buffer. You can reedit it, but it will not be stored permanently on diskette until you enter a save or write command while in the Command Processor.

PIE COMMAND PROCESSOR

You are transferred to the Command Processor after choosing the first option in the System Menu, PIE TEXT EDITOR; or after exiting the PIE Text Editor with a **CTRL-SHIFT-P** (or **CTRL-@**) command; or when returning to PIE from FORMAT. The Command Processor is used to handle the loading and saving of all PIE Writer files on diskette and to provide additional file-manipulation and input/output capabilities.

The Command Processor has three types of commands: Control, Input/Output, and Auxiliary. All commands must be followed by a **RETURN** and are entered in response to the Command Processor prompt:

```
COMMAND?:
```

Prior to pressing **RETURN**, you may delete an entry by typing **CTRL-X**. For users with an Autostart ROM, the Command Processor also supports cursor moves and screen copying using **ESC I-J-K-M** as described in your the reference manual for your computer.

Control Commands

CP Command	Function
N(EW)	Edit new file; clears current file
E(DIT) n	Re-edit current file (optionally starting at line 'n')

F(ORMAT)	Transfer to FORMAT program
C(ATALOG)	Catalogs current drive
?	Display remembered filename (fn), set with L fn, S fn, <fn,>fn
LE(NGTH)	Display remembered filename, length, and available memory remaining
CTRL-Y	Enter system monitor; **CTRL-Y** returns you to Command Processor
CALL addr	Calls machine language routine at addr; may be signed decimal or $hex

Parentheses around part of a command in the above table indicate that typing the enclosed part is optional.

N (or **NEW**)—Clears the edit buffer before beginning a new text file. The **N(EW)** command is followed by the precautionary prompt OK TO CLEAR?: A **Y** (for yes) response produces the empty text widow; an **N** (for no) redisplays the CP prompt.

E (or **EDIT**) **n**—Returns you to the text editor without clearing the previous contents of the buffer, so that you can begin editing a file or continue inputting text. This command must be used after loading a file from diskette for re-editing.

The cursor appears at the beginning of the text for a recently loaded file; otherwise, the cursor reappears at its previous position if you have worked with this file and have transferred to the Command Processor temporarily (for example, to update the diskette file).

Entering an optional 'n' value allows you to specify the line number of the file you would like the cursor to appear at the beginning of. If the line number is greater than the highest line number in the text buffer file, the cursor appears at the beginning of the file's last line. (The command **E $** will also jump to the end of a file.)

F (or **FORMAT**)—Transfers you to the FORMAT Text Processor, which formats a file for output to the printer, screen, or diskette.

C (or **CATALOG**)—Displays the filenames on the currently active diskette. The slot, drive and volume number of the disk drive whose catalog you want displayed may optionally be specified if more than one drive is being used.